SOLID-MEDICINE CABINET AND INVENTORY SYSTEM WITH TIME BASED ALARM AND LIGHT EMITTING DIODE (LED) NOTIFIER

by

Catalan, Neil Carlo P.
Jose, Josiah David D.
Leandicho, Carla Louie H.

A Design Report Submitted to the School of Electrical Engineering, Electronics Engineering, and Computer Engineering in Partial Fulfilment of the Requirements for the Degree

Bachelor of Science in Computer Engineering

Mapua Institute of Technology
February 2012

Approval Sheet
Mapua Institute of Technology
School of EECE

This is to certify that I have supervised the preparation of and read the design report prepared by **Catalan, Neil Carlo P., Jose, Josiah David D., Leandicho, Carla Louie H.** entitled **SOLID-MEDICINE CABINET AND INVENTORY SYSTEM WITH TIME BASED ALARM AND LIGHT EMITTING DIODE (LED) NOTIFIER** and that the said report has been submitted for final examination by the Oral Examination Committee.

Dionis A. Padilla
Design Adviser

As members of the Oral Examination Committee, we certify that we have examined this design report, presented before the committee on **February 14, 2012**, and hereby recommended that it be accepted in fulfilment of the design requirements for the degree in **Bachelor of Science in Computer Engineering.**

Ernesto M. Vergara Jr.
Panel Member

Jerry V. Turingan
Panel Member

Lilibeth D. Mendoza
Chairman

This design report is hereby approved and accepted by the School of Electrical Engineering, Electronics Engineering, and Computer Engineering in partial fulfilment of the requirements for the degree in **Bachelor of Science in Computer Engineering.**

Felicito S. Caluyo
Dean, School of EECE

ACKNOWLEDGEMENT

We are sincerely thankful to our adviser, Dionis Padilla, whose encouragement, guidance and support from the initial to the final level enabled us to develop an understanding of the subject.

It is a pleasure to extend gratitude to those who made this design project possible such as our parents who gave us the moral support, and our friends, Rommer Cañete and Francis Evangelista who helped us in the programming part of this design project. We also would like to make a special reference to Ms. Ayra Panganiban who is our professor in design course. Without her guidance, we could not have completed this design project. We also like to thank our design panels for giving us the necessary corrections in our documents.

Lastly, we offer our blessings to everyone who supported us in any respect during the completion of the project.

TABLE OF CONTENTS

TITLE PAGE		i
APPROVAL SHEET		ii
ACKNOWLEDGEMENT		iii
TABLE OF CONTENTS		iv
LIST OF TABLES		vi
LIST OF FIGURES		vii
ABSTRACT		viii
Chapter 1:	DESIGN BACKGROUND AND INTRODUCTION	1
	Overview	1
	Customer	2
	Need	2
	Solution	3
	Benefits	8
	Definition of Terms	9
Chapter 2:	REVIEW OF RELATED DESIGN LITERATURE & STUDIES	12
	Medicine Dispensers	12
	Microcontroller Unit	23
	Serial Communication from Microcontroller to Computer	25
Chapter 3:	DESIGN PROCEDURES	27
	Hardware Development	27
	Software Development	31
	Prototype Development	37
Chapter 4:	TESTING, PRESENTATION, & INTERPRETATION OF DATA	39
Chapter 5:	CONCLUSION AND RECOMMENDATION	44
	Conclusion	44
	Recommendation	45

References 47

Appendices 48

A. Operation's Manual 48

B. Pictures of Prototype 77

C. Data Sheets 82

D. Program Listing 101

E. Price List 135

F. Letter of Intent 136

G. IEEE Format Article of the Design 137

LIST OF TABLES

Table 4.1 Expected Action by the MCU with the corresponding input data

Table 4.2 Test for LED Responses on their Respective Drawers

Table 4.3 Test for Buzzer Response

Table 4.4 Test Inputs for the Software Application

Table 4.5 Test for LED Responses on their Respective Drawers with C# code

Table 4.6 Test for Buzzer Response with C# code

LIST OF FIGURES

Figure 2.1: Programmed Medication Dispenser

Figure 2.2: Side-view of Programmed Medication Dispenser

Figure 2.3: Cabinet for Dispensing Medicines at Predetermined Times

Figure 2.3: General Circuit of the McLaughlin's Invention

Figure 2.4: Perspective of the Portable Medicine Cabinet with Timer

Figure 2.5: Front, Side Elevation and Side View of Medicine Cabinet with Timer

Figure 2.6: The Smart Medical Refrigerator

Figure 2.7: Paper Prototype of a Human-Centered Design of Medicine Dispenser

Figure 3.1: Conceptual Framework

Figure 3.2: Block Diagram of Solid-Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode (LED) Notifier

Figure 3.3: Schematic Diagram

Figure 3.4: Program Flow for the Software Application

Figure 3.5: Data Flow Diagram for Scheduling

Figure 3.6: Data Flow Diagram for Inventory

Figure 3.7: Program Flow for PIC16F877A

ABSTRACT

The common causes of medication errors are missing doses, taking incorrect amounts and taking medicines at the wrong time. These mistakes could lead to increase discomfort, inadequate diseases prevention and possibly even death of the patient. The main purpose of this study is to lessen the medication errors and cost of the hospitals by designing a medicine cabinet. The medicine cabinet has light indicators in each drawer and an alarm that will notify a nurse if a particular patient needs medication. It has also inventory system to monitor the patient's medicine. It will be developed using a MCU (MicroController Unit), a high level programming language and a database management system. The MCU is connected to several LED (Light Emitting Diode), buzzer and to the computer. We intended to have two tests, which are the LED and alarm testing using UART terminal and the C# code

Keywords**:** UART, MCU, LED, Medicine cabinet, Inventory system

Chapter 1

DESIGN BACKGROUND AND INTRODUCTION

Overview

Hospitals are one of the facilities that are used by people to give them medical, surgical, or psychiatric treatment and nursing care. It is important to ensure the safety and security of its patients by giving them right medication, healthy food, and clean environment. By considering safety and security, the patient will gain trust to the service of the hospital. Nurses are of big help in a hospital because they are assigned at the nurse station in each ward where they manually check the schedule of patient's time for taking their medicines as ordered by a doctor. The common problems that the nurses commit include having trouble with the patients' schedule and what medicines they need to deliver to their patients. It is critical for the nurses to give the right medicine at its scheduled time of medication to ensure the patient's safety and health.

Nowadays, hospitals are using different technologies in medication to ensure the safety of its patient. Before, hospitals are using medicine cabinet that has compartments for containing supplies of different kinds of medications to be accessed by a healthcare attendant for preparing individual medication dosages for named patients. With the use of technology, some of the medicine cabinets include a processor having a memory for storing the names of patients and their prescribed medication dosages, and a display screen for displaying the patient

names and their respective prescribed medication dosages. The tray includes a display screen for displaying the patient names and their respective medication dosages, and a communication link with the medicine cabinet through which the cabinet processor communicates to the tray the patient names and their respective medication dosages. Some of the medicine cabinets have an alarm that will sound reminding the user when medicine shall be taken, and making the correct dose available. The advantages of this automated medication cabinets is that it will lower costs associated with pharmaceutical distribution, monitors inventory, further reduction of errors, and relieving professional pharmacists and nursing personnel of many tasks.

Customer

Our target customer is Tondo Medical Center located at North Bay Boulevard, Balut Tondo, Manila. It is a 200-bed capacity tertiary public medical center established in 1971 by virtue of Republic Act no. 6375. It presently operates under the supervision and control of the Department of Health (DOH).

Need

In a Philippine hospital usually a particular nurse is assigned at the nurse station in each ward where he/she manually checks the schedule of patient's time for taking their medicines as ordered by a doctor, and usually a group of patients is assigned to a nurse, the reason for this is to cut the cost of expenditures for nurses. This scenario is evident in government hospitals which lead to an error on the part of the nurses. The assigned nurse is the one who is

responsible for preparing the medicine requirement of all the patients he/she handles in a ward (for example a medical ward). Also, the assigned nurse checks the stocks of medicines in the cabinet. With this situation, a need for a cabinet for solid medicines with time based notifier is required so that the assigned nurse will not be confused on what slot in the medicine cabinet he/she will open. If the time comes for taking a medicine, the door of the cabinet will have an LED indicator then an inventory application in a computer will check what solid medicines has to be taken out by the assigned nurse and it will check how much medicine is available, thus simplifying the work of the assigned nurse.

Solution

With the presented problem, the designers came with a solution to design a solid-medicine cabinet and inventory system with time based alarm and light emitting diode (LED) notifier. First it will implement an application for inventory system that will monitor what medicine should be taken out and check the quantity of available medicine. And lastly is to notify nurses on the schedule of patients on his/her medicine intake using an alarm and LED as light signals on each slot in the cabinet.

The medicine box or shelf which will be comprised of 16 drawers with 16 LEDs beside each drawer particularly at the right position, each drawer has 2" x 4 ¾" x 1 ¼" dimensions. `

Constraints

The constraint/s of the solution in terms of economics is first the solution requires a good budget in order to install in a particular hospital. The designers of the solution want to have a plan for actual implementation, but still the return of investment is not yet considered.

In terms of manufacturability, the solution may require a custom built cabinet for medicine and the electronic parts which may lead to larger cost compare with an ordinary cabinet. And lastly, a computer is needed for the design solution to work.

In terms of capability, the medicine cabinet is only intended to store solid medicines for oral medication. The computer is not capable of accessing other medicine cabinet. Thus, it will provide monitoring on one cabinet only.

In terms of sustainability, the medicine cabinet will use a regular adapter that is plug in to a regular outlet so sustainability in power source is not yet considered. In addition, the medicine cabinet is interfaced with the computer with database which requires maintenance by an expert.

In terms of software application, the solution will require initial inputs on the stocks of medicine, patient's name, name of the doctor, time of intake, slot number on the cabinet, the name of medicine and etc. Next, only the authorized person can assign the schedule for each nurse on the computer. Third, the connection between the cabinet and computer is not wireless. Fourth, the software application will use the system clock of the computer. And lastly the

software will not recognize if the nurse has taken out the exact quantity of medicine, the quantity of medicine could be more than or less than the required number of medicine.

In terms of security, the hardware and software application provide a low level of security. Because the designers believe that security is another scope of the design, futher studies on it will not be included with the scope of this solution due to time constraints.

Impact

The impact of the design in terms of meeting desired needs to health and safety is that the design will help the nurses to provide the medication for the patient in proper time, and they will not be confused because the design will provide light on LEDs on the slots intended for those patients that are scheduled to intake their medicine in the cabinet.

Differentiation

Manual method

Almost all hospitals in the Philippines uses a manual method for keeping medicines and sometimes this method brings a possible cause of accident, like for example a nurse which gave wrong medicine to a patient.

Difference: Our design proposal lessens the possibility mentioned above by including a light signal; there will be a slot on the medicine cabinet for the LEDs. When the medicine needed is inside the drawer, the assigned LED for the said drawer will light up; otherwise, it will not. Moreover, the proposed system

provides an organize way to remind an individual in the hospital for the patient's scheduled medication. The proposed system also keeps track of the medicines that are being stored and taken out to the patient's assign drawer, with the use of the inventory software.

A Medicine Box Prompter Kit Using RFID

Published on June 2008 at Mapúa Institute of Technology

The main key in this design is by the use of radio frequency in alerting the patient once he or she is out of range, which effectively reminds him or her to carry important medication before leaving home.

Difference: Our design will be implemented to a hospital particularly at the medical ward unlike the said project above it is only for a single patient that is capable of operating the apparatus to take their medication on time. It also uses RFID technology to remind the patient, while in our design we will use microcontroller to control the LED to notify the nurse for patient's medication.

Medication Dispensing System including Medicine Cabinet and Tray therefore

Inventors: Haitin, David and Asseo, Gilead

A medication dispensing system includes a medicine cabinet having a plurality of compartments for containing supplies of different kinds of medications to be accessed by a healthcare attendant for preparing individual medication dosages for named patients; and a tray having a plurality of sections for receiving a plurality of receptacles each adapted to contain one or more medication dosages prescribed for a named patient. The medicine cabinet

includes a processor having a memory for storing the names of patients and their prescribed medication dosages, and a display screen for displaying the patient names and their respective prescribed medication dosages. The tray includes a display screen for displaying the patient names and their respective medication dosages, and a communication link with the medicine cabinet through which the cabinet processor communicates to the tray the patient names and their respective medication dosages.

Difference: Our design project can differentiated from the said invention which requires a construction of cabinet processor while our design does not need it. In our design solution we will not include tray with multiple compartments unlike the design mention above. Our design solution will use a computer for its inventory of medicine while the design mention above will use a processor having a memory for storing the names of patients and their prescribed medication dosages. So, with the manufacturability of our design much more simple than one mentioned above.

Cabinet for Dispensing Medicines at Predetermined Times

Inventor: McLaughlin, John T.

A cabinet containing individual compartments each with an individual time lock for holding various doses of prescribed medicines respectively is positioned adjacent a given patient's bed. The individual time locks are programmed to open at given times during a 24 hour period at which the medicine in the corresponding compartment is to be given to the patient. A

signal light advises a nurse whenever any one of the compartments is unlocked. It is thus assured that the correct dose of the correct prescribed medicine is given to the correct patient at the correct given time for that particular medicine.

Difference: Our design project can also differentiated to this design because the said invention is positioned adjacent to a given patient while our project is located at the medical ward nurse station and our design is only intended for solid medicine only. Moreover, our design does not provide a lock one mentioned above, but our design solution provides a light indicator for each slot in the cabinet. Our design solution also implements an inventory system that will keep track on what medicine is disposed from the cabinet in which an added features than the one mentioned above.

Benefits

The benefits of the proposed system are as follows: First, the checking of patient's medicine schedule is automated. Next, it provides safety in picking the medicines. And it will tell on what medicines was being disposed to patients. In short, it will serve as an electronic guide to the nurse thus; simplifying his/her effort which will make his/her do another activity inside the hospital related to his/her work.

Definition of Terms

LED – a light-emitting diode (LED) is a semiconductor device that emits visible light when an electric current passes through it and serves as light signals on each cabinet.

Inventory System – the system that will monitor what medicine should be taken out and check the quantity of available medicine.

Solid Medicine Cabinet - a cabinet which consists of 16 drawers with 16 led's and also the allow medicine to be store is only solid.

Time Based Alarm - a buzzer that will notify the nurses on the schedule patients on his/her medicine intake.

RFID (Radio Frequency Identification) – a technology that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency (RF) portion of the electromagnetic spectrum to uniquely identify an object, animal, or person.

Nursing Care – described as a care given to patients by nursing service personnel.

Medical Ward – block forming a division of a hospital (or a suite of rooms) shared by patients who need a similar kind of care.

Smart medical refrigerator – a medical dispenser that monitors whether the patient took his/her medication.

AphA (American Pharmacists Association) - previously known as the American Pharmaceutical Association, founded in 1852, is the first-established professional society of pharmacists within the United States.

MCU (Microcontroller Unit) – a single chip that contains the processor (the CPU), non-volatile memory for the program (ROM or flash), volatile memory for input and output (RAM), a clock and an I/O control unit.

RS-232 (Recommended Standard 232) - the traditional name for a series of standards for serial binary ended data and control signals connecting between a *DTE* (Data Terminal Equipment) and a *DCE* (Data Circuit-terminating Equipment).

(TTL) Transistor–transistor logic – a class of digital circuits built from bipolar junction transistors (BJT) and resistors. It is called *transistor–transistor logic* because both the logic gating function (e.g., AND) and the amplifying function are performed by transistors (contrast with RTL and DTL).

MAX232 – integrated circuit that converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.

I²C (pronounced I-squared-C) – created by Philips Semiconductors and commonly written as 'I2C' stands for Inter-Integrated Circuit and allows communication of data between I2C devices over two wires. It sends information serially using one line for data SDA (Serial Data Line) and one for clock SCL (Serial Clock line).

LPT (Line Print Terminal) – the original, and still common, name of the parallel port interface on IBM PC-compatible computers. It was designed to operate a text printer that used IBM's 8-bit extended ASCII character set.

Chapter 2

REVIEW OF RELATED DESIGN LITERATURES AND STUDIES

Medicine Dispensers

Programmed Medication Dispenser, 1968

In the year 1968, John Glucksman's and Joseph R. Kub's invention entitled "Programmed Medication Dispenser (see Figure 2.1 and 2.2)" was patented. "The invention relates to a timed dispenser device for dispensing at pre-selected time intervals, any desired dosage, such dispensing action being accomplished by providing a delivery station communicable with and normally isolated from storage means for holding a plurality of doses and utilizing a timer to actuate control means which provide communication at selected intervals and sensing means responsive to the passage of a dose to the delivery station to deactivate the timer and thereupon to reactivate the timer when the dose is removed from the delivery station" (John Glucksman and Joseph R. Kub, 1968).

The said invention aims to have four objectives one was to provide a timed dispensing apparatus for medications or the like. Second, was to create of an automatic device in which can be loaded and if in loaded condition will positively prevent access to the contents. Third, was to create a device that has sensor for detecting the delivery of a dosage to the delivery station, the sensor deactivate the timer device if there is a dosage and hence prevent the deposit of

a further dose unless the prior dose has been physically removed from the dispensing station. And lastly, was to create a device that readily set up to deliver dosages at selected intervals, the timer mechanism for varying the dispensing intervals must be flexible so that the dosage schedule may be easily varied even by an unskilled operator.

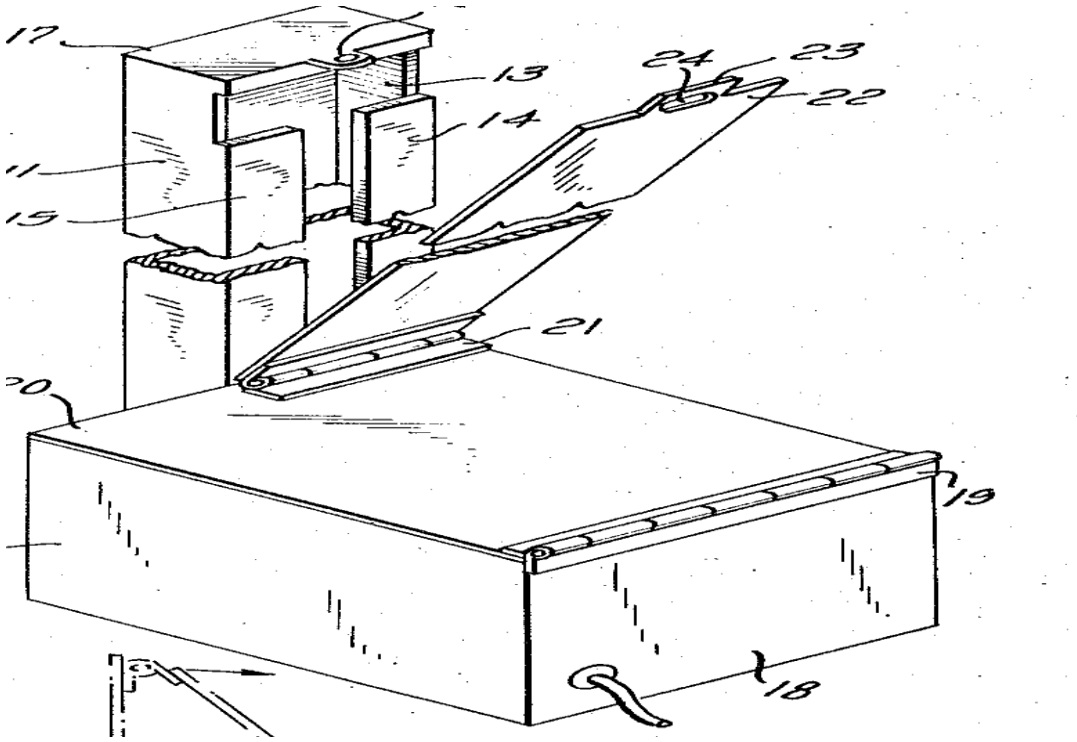


Figure 2.1: Programmed Medication Dispenser

Source: John Glucksman and Joseph R. Kub, 1968

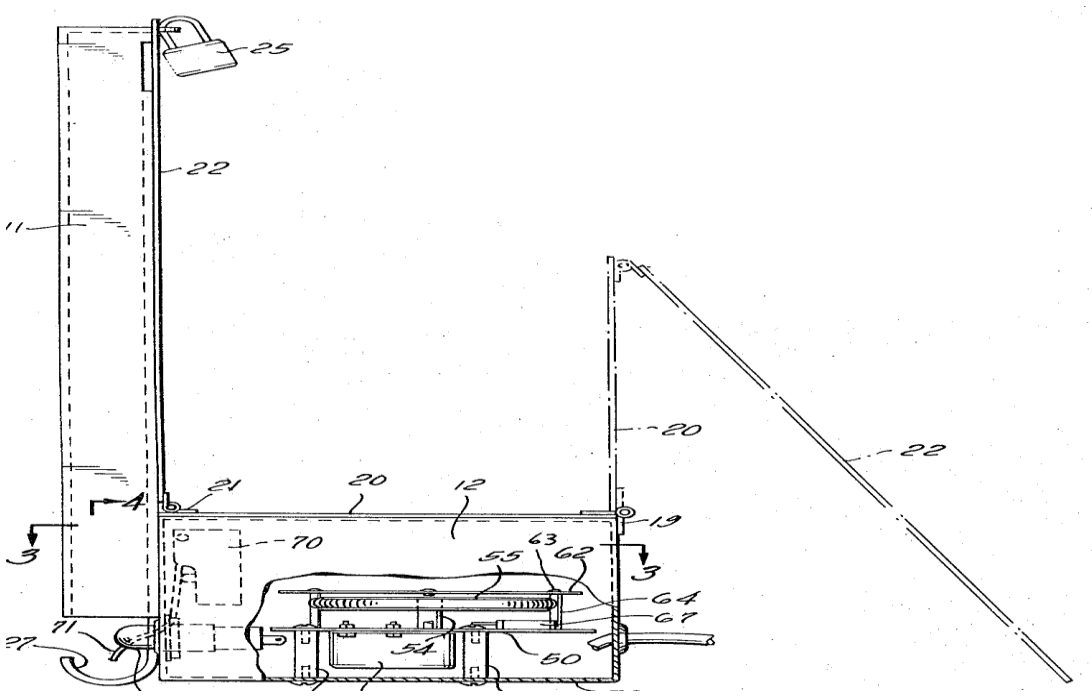


Figure 2.2: Side-view of Programmed Medication Dispenser

Source: John Glucksman and Joseph R. Kub, 1968

Cabinet for Dispensing Medicines at Predetermined Times, 1973

In the patent paper of John T. McLaughlin, "Cabinet for Dispensing Medicines at Predetermined Times" the invention was created to minimize the common errors in carrying out medicines to the patients. They used a cabinet to create individual compartments for each patient. Each compartment has a time lock and a signal light. The signal light was created to give the nurse an idea what compartment the nurse can get the prescribed medicine for the given patient. The time lock was used to give the prescribed medicine to the patient at

the correct time (see Figure 2.3 to have an idea on the appearance of the said invention).

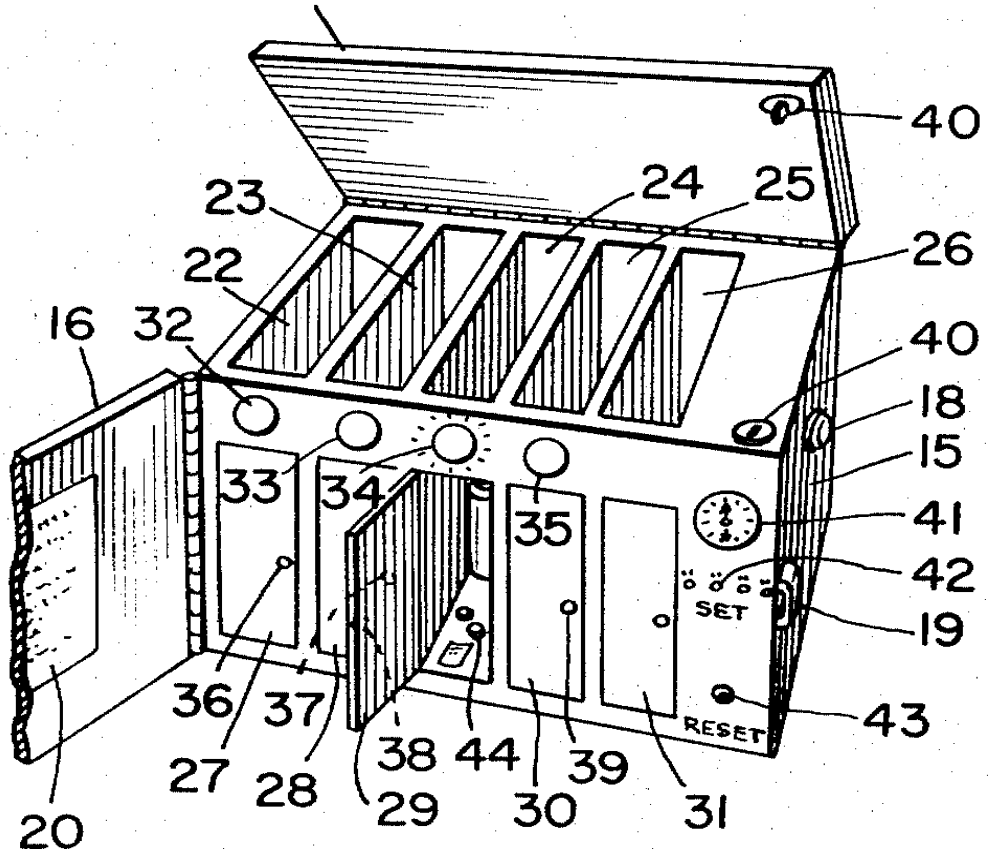


Figure 2.3: Cabinet for Dispensing Medicines at Predetermined Times

Source: John T. McLaughlin, 1973

The invention was used to ensure that the correct dose of medicine will be given to the right patient. The time of medication of the patient was programmed. When it is time for the patient to take its medication, the lock will be unlocked based on the programmed time and the signal light will notify that the nurse that it is time for the patient to take its medication. The individual lock for each compartment was programmed according to the schedule of the patient. An output signal is passed to a counter and an "OR" circuit (see Figure

2.3), the output signal also passes through the master clock and the main signal light that the program actuates to unlock the associated compartment door. Another synergizing signal will also be passed to the nurse station to inform the nurse that it is time for medication of the patient. Setting the time for the schedule is set by using a counter. The clock pulses determines the time for the compartment to open.

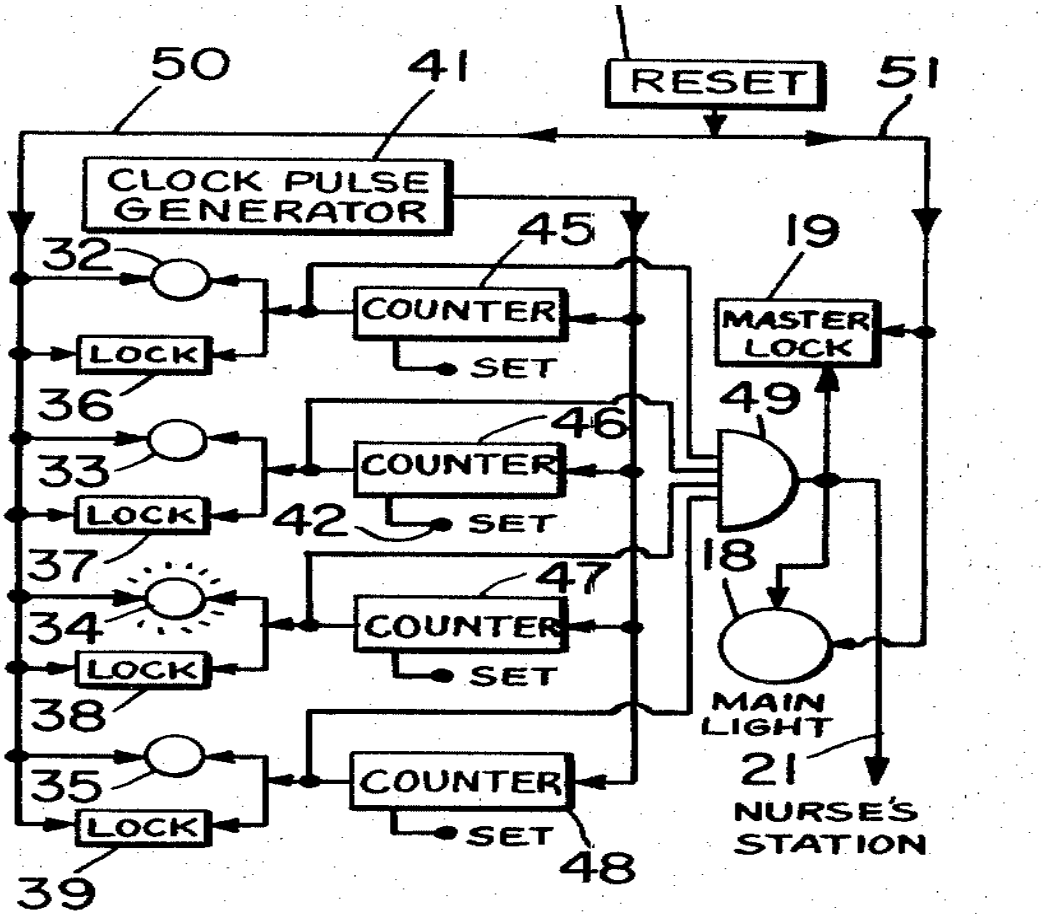


Figure 2.3: General Circuit of the McLaughlin's Invention

Source: John T. McLaughlin, 1973

Portable Medicine Cabinet with Timer, 1979

In the year 1979 particularly 4th of June, Thurmond Hicks and Brenda Hicks filed a US patent for their invention "Portable Medicine Cabinet with Timer". The purpose of their invention is to assure that an individual does not both become confused and forget to take doses of certain prescribed medicines, or take doses of the prescribed medicine too often so as to endanger health (Hicks T.A. and Hicks B. G., 1981).

The portable medicine cabinet was only intended for private use by an individual taking several different medicines, in which may be carried from/to the home, office, and the like. It comprises a plurality of freely accessible compartments and time computer into which a plurality of predetermined time intervals for taking a dose of the respective medicines may be entered (Hicks T.A. and Hicks B. G., 1981). An indicator is electrically connected to the time computer which indicates the predetermined time intervals for the medicines should be removed from its respective compartment.

Thurmond Hicks and Brenda Hicks stated in their paper in the year 1981 that many individuals during that year are taking several prescription drugs the same period of time and it is believed that a problem exists in keeping track of the predetermined times for taking a dose of the respective medicines, in which we can say that is still evident even today. In fact according to American Pharmacists Association (AphA) through the report of Institute of Medicine report in 2006 that 4 of 5 adults take Rx, OTC, or dietary supplement every week, 33%

take 5 or more medications weekly, 1.5 million injured by medication errors, and lastly medication errors cost $3.5 billion.

Therefore, it can be understood that an object of the invention is for the inventors to provide a portable medicine cabinet with a timer which may be used by individuals who are taking several medicines at one time and desires aid in keeping track of the time that a dose of each medicine should be taken. Below are the drawings of the said invention.

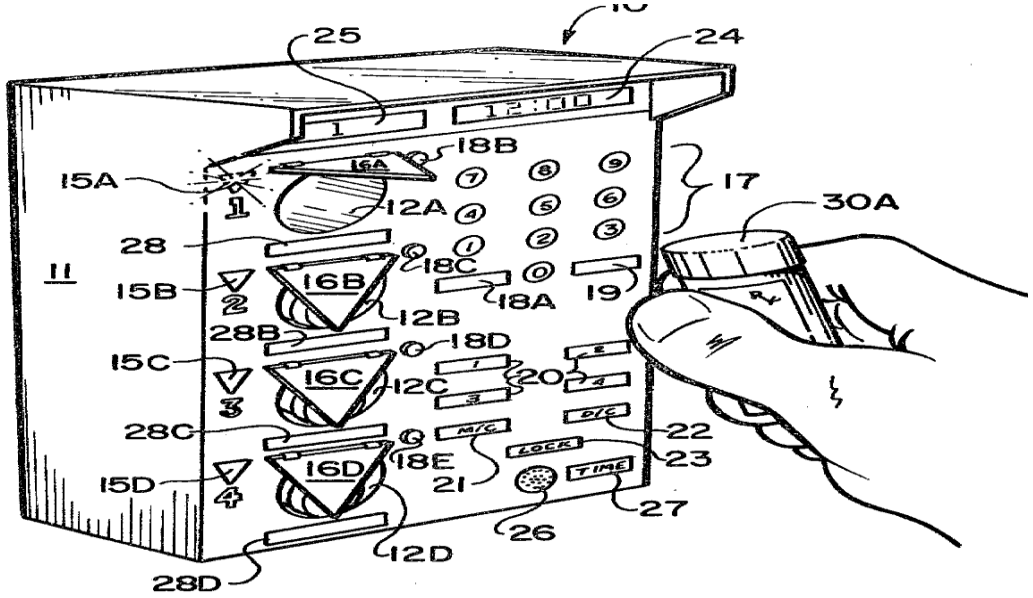


Figure 2.4: Perspective of the Portable Medicine Cabinet with Timer

Source: Hicks T.A. and Hicks B. G., 1981

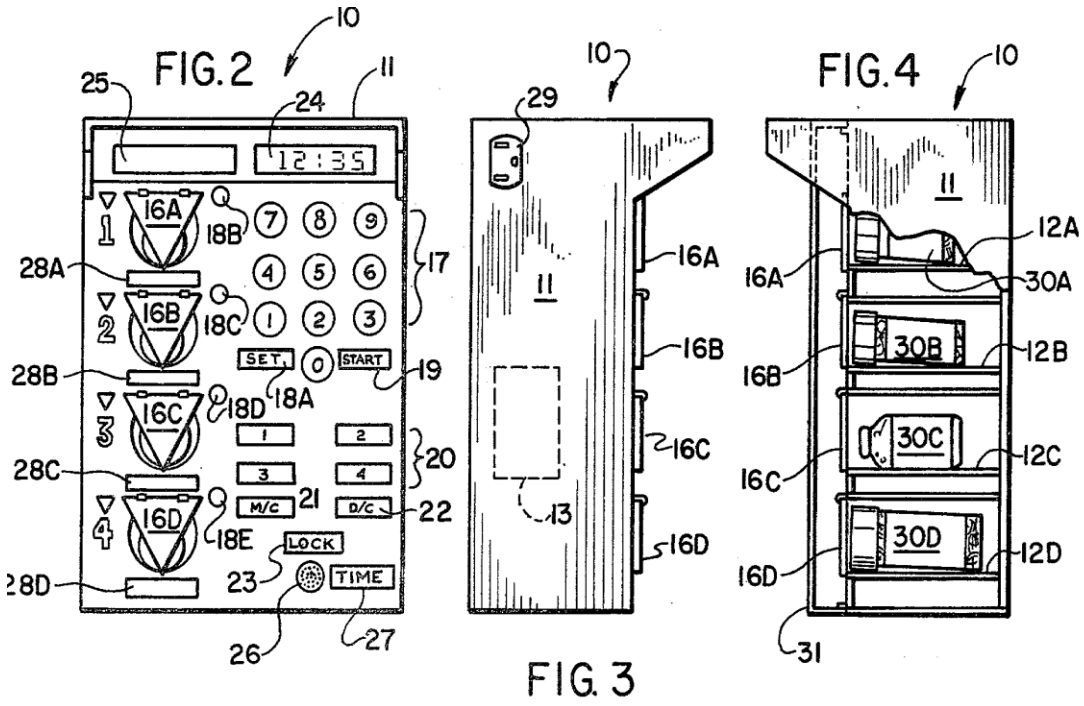


Figure 2.5: Front, Side Elevation and Side View of Medicine Cabinet with Timer

Source: Hicks T.A. and Hicks B. G., 1981

Figure 2.4 is a perspective of the portable apparatus for storing a plurality of different medicine bottles and indicating at certain predetermined times that a dose of medicine should be taken.

Figure 2.5 FIG 2 is a front elevation view of the portable apparatus of invention with the medicine secured within the respective compartments therefore; FIG 3 is a side elevation view of the portable apparatus of the invention showing the time computer means in phantom lines within the cabinet; and FIG 4 is a side view, partly in section, of the portable apparatus of the present invention with the snap-on cover.

The Smart Medical Refrigerator, 2005

In the article titled "The Smart Medical Refrigerator" by Paul Kuwik, Thomas Largi, Matt York, Dennis Crump, David Livingston and James C. Squire they developed a device (see Figure 2.6) that will help elderly diabetics that were living alone without a nearby family. The smart medical refrigerator is a medical dispenser that monitors whether the patient took his/her medication. An alarm and a blinking LED will alert the patient if the patient needs to take the medication. The schedule is programmed in a Microcontroller. It can be reprogrammed using an externally accessible serial adapter for any change in the medicine's dosing schedule. It has a modem that dials to an internet service provider (ISP) because whenever the patient did not take his/her medicine, it establishes a simple mail transfer protocol (SMTP) connection to the mail server, and sends an email to any designated contacts such as doctor, a family member, and friends to alert them that the patient did not take the medication. Opening the door was used like an interrupt that will be sending a signal to the microcontroller that the patient had taken the medication. When the door was closed during the time of the medication it will send alerts to any designated email address. This device can also operate during blackouts. While the device is connected to the house current, it will be automatically recharged to maintain its peak energy. It has also a battery level indicator that is used when the battery is in backup operation. The battery indicator uses Light-emitting diodes (LEDs) to display the remaining life of the battery.

Figure 2.6: The Smart Medical Refrigerator

Source: Paul Kuwik, Thomas Largi, Matt York, Dennis Crump, David Livingston and James C. Squire, 2005

During the testing process, the problem that they encountered is that the when the door of the device was not shut. Overall the patient found the device useful. This device acts like an active-sensor that requires patient interaction to send out an alarm.

Enhancing Medicine Adherence through Multifaceted Personalized Medicine Management, 2011

In a research entitled "Enhancing Medicine Adherence through Multifaceted Personalized Medicine Management", it was said that medicine adherence is a growing problem that affects not only patients but also the healthcare industry (Min-Hui Foo, Jit Chee Chua, Jamie Ng, 2011). The paper

presents a brief overview of the severity of medicine non-adherence, especially to elderly patients of chronic conditions, and the reason behind the phenomenon. With that the research focuses on the development of a medicine management system in the form of medicine dispenser and a web portal which attempts to bring under a single platform to many-sided approach to address medicine adherence and the benefits that ensue.

The medicine dispenser stores pills which reminds patient when it is time for medication and dispenses the right dosage to the patient. The web portal receives and stores medication data from the medicine dispenser. The automatic reminders and correct dispensing of dosage by the medicine dispenser serve to increase the patient's confidence in his or her ability to follow the medication regimen. Both the medicine dispenser and web portal would incorporate the function of providing medicine information thus, educating the patient to better understand their condition and drug therapy. With better knowledge about his or her condition and the intended effects of the medications, intentional non-adherence and denial of their condition can be subverted. It also enables both healthcare professional and patient to access the patient's medicine intake history, which provides both party with insights and better understanding of patient's medicine taking patterns. Below is the figure of the medicine management system that includes a medicine dispenser and a web portal.

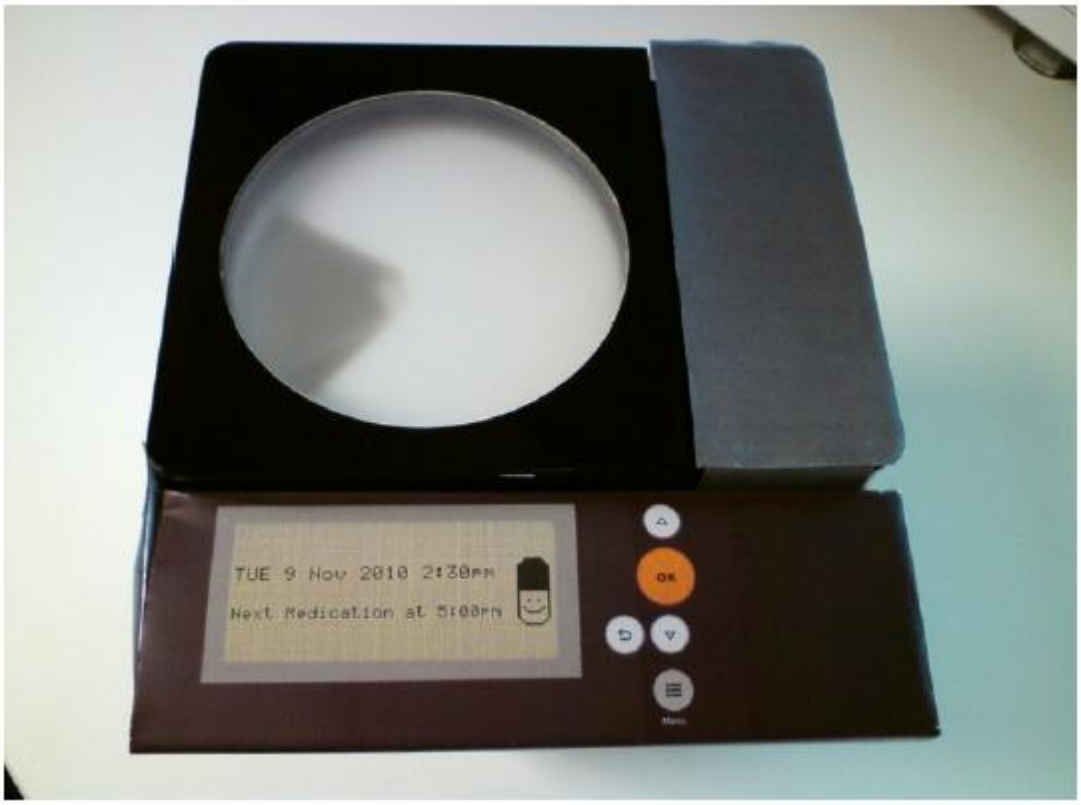


Figure 2.7: Paper Prototype of a Human-Centered Design of Medicine Dispenser

Source: Min-Hui Foo, Jit Chee Chua, Jamie Ng, 2011

Microcontroller Unit

In the article of Wolfgang Abele and Markus Hofmann, "Driving LEDS with a PIC Microcontroller" different applications were tried that can be tested to a LED using a PIC Microcontroller. One of the tests that they conducted is the failure recognition. First is that they explained the different parts of the microcontroller. It stated that the pins of the microcontroller can be configured as analog input. The analog voltages can be converted to digital values by means of an A/D module. The signal to be converted is chosen by software, and

forwarded to the A/D module. This module requires reference voltage. The voltage can either be applied externally at an input pin, or internally via so-called voltage reference module. In the latter case, the supply voltage to the PIC must be kept constant by means of a voltage regulator. For the programming part of the PIC microcontroller, software is used for creating assembler programs which are downloaded and executed in the microcontroller. The use of C compiler is recommended to easily manage and permit changes to readily make.

In this article, they used a matrix connection with a resistor on each LED. Each LED possesses its own series resistor. These resistors can be used as reference resistors for adjusting the current in the diode. If one LED fails, the remaining LEDs continue to function. When circuit is in a series connection, a failure would lead to more current in the parallel connected diodes and an uneven distribution of brightness would arise.

For the failure recognition, they test individual LEDs within an array for total failure. When an LED leads to a break in path, it causes no current to flow through the series resistor. The proposed solution here is to use a multiplexer. The multiplexer can be digitally controlled by the microcontroller. Each LED is assigned an address which allows it to be selected by the PIC, in order to determine the current passing through the diode for current regulation or to assist in failure recognition.

Serial Communication from Microcontroller to Computer

The article "Concepts for transmitting data from a PC to a microcontroller" by Ryan Kittredge discussed a serial port interface. An RS232 serial port can be used in transmitting data between PC and the microcontroller. There are many functions of serial one of the functions is that we can use it for downloading programs that have been compiled on a microcontroller. There are drawbacks when using an RS232 because it uses a negative logic. The microcontroller uses standard TTL logic so the RS232 signal has to be sent through another device to convert the negative logic back to TTL. A MAX232 chip can be used to convert the negative logic to TTL and keep the data in serial format. They also discussed the I^2C which is a type of serial communication developed by Philipps. This technology is widely used in all kinds of electronic devices. It is a two wire bus, the first wire is the SCL or the Serial Clock line and the other is the SDA or the Serial Data Line. The two wire bus allows data transmission between multiple devices. Parallel port interface is also discussed in this article. There are many benefits in using a parallel port, one of which is that it is easy to manipulate with the software and the data is transmitted using standard TTL 0-5v signals. Another benefit is that there is no need to put the signal back together. A parallel port has a 25 pin connecter on the computer that is commonly known as printer port, LPT1 or LPT2. In utilizing the parallel interface, 8 pins are used for data transmission; 1 pin that signals that the data is ready and clocks it through, 1 pin to signal whether the data transmission was

an address or actual data, and 1 pin to send a signal back to the PC telling that the current task has been accomplished.

Chapter 3

DESIGN PROCEDURES

This chapter gives detailed information about the procedures used in developing the design entitled Solid-Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode (LED) Notifier. Software and Hardware Development are discussed thoroughly in this chapter to guide future researchers who aspire to innovate the said design.

Hardware Development

This section provides the overview of the steps that the designers undergo in developing the hardware of the said design. Hardware Development discusses the components used and the flow of data through the components in terms of conceptual framework, block diagram and schematic diagram. Major compositions of this design are discussed in terms of block diagram and schematic diagram to see the actual circuit comprising of each diagram.

First, the designers researched and gathered information about previous related designed devices as a basis for circuit design. Second, the designers chose the right kind of components, then the designers started to simulate the circuit, after getting the correct output the designer began to develop the said design. Finally, the designers proceed to testing. While conducting the testing, some errors were encountered; in this case, the designers proceed in debugging the device.

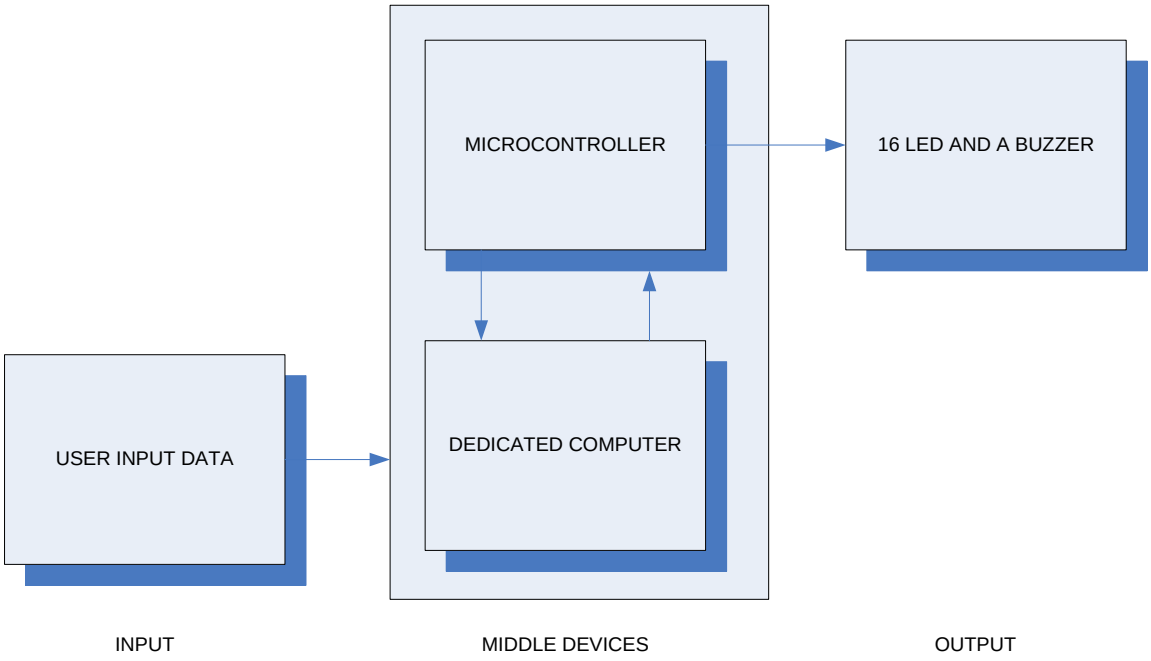


Figure 3.1 Conceptual Framework

Figure 3.1 is the conceptual framework of the proposed system. The designer used the computer as a middle device that accept inputs from the user and send a specific command to the microcontroller which serve as a middle device as well. The computer also monitors the schedule of the patient's which helps the nurses to their tasks. The computer will also have inventory software that will monitor the quantity of the medicines of the patients. The microcontroller will serve as the driver for the buzzer and the sixteen LEDs.

The researchers constructed a block diagram that illustrates the flow of data in the design of the system which is shown in Figure 3.2.

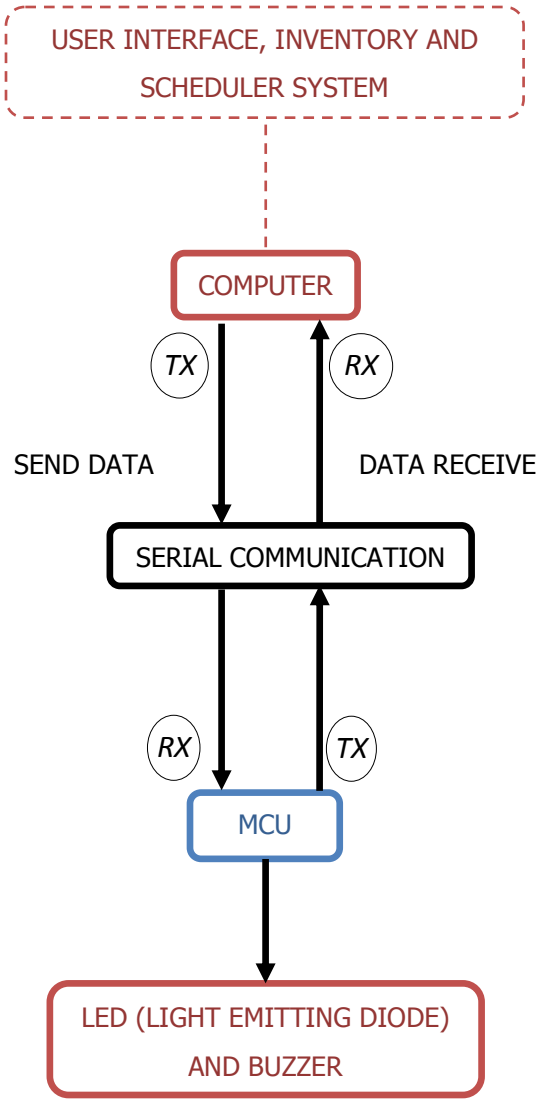


Figure 3.2 Block Diagram of Solid-Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode (LED) Notifier

Figure 3.2 shows the block diagram of the solution; it clearly shows that a computer with the software application is interfaced to the MCU; in this case, the designers used PIC16F877A. The computer and the MCU communicate with each other using serial communication; the designers used the USB-to-RS232 converter cable for laptops and netbooks to provide serial connection since most of the laptops/netbooks don't have serial port.

The buzzer and the 16 LEDs are interfaced to the output ports of the MCU. The communication between the computer and the MCU can be tested using a terminal or programmatically.

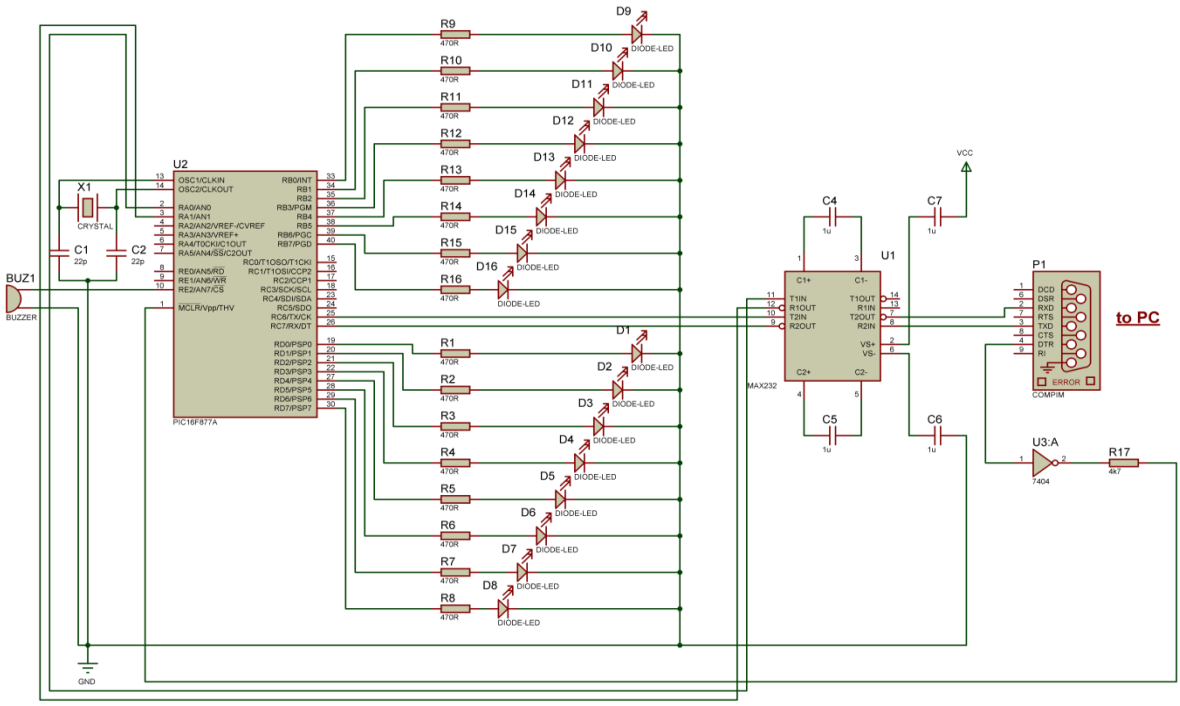


Figure 3.3 Schematic Diagram

Figure 3.3 shows the Schematic diagram for the said solution; LED number one to eight is attach to the port D of PIC16F877A while LED number nine to sixteen is attach to port B of the said MCU. Each LED has 470 Ohms resistor in series to them for the protection of each LEDs. The buzzer is attached directly to port E pin number 2 of the said MCU. Since the said model of PIC doesn't have internal clock the designer provide one, using a 16MHz crystal together with the two bypass capacitor. The designer adapted the setup of this external clock to other existing circuits because this is the common setup for an

external clock with this kind of MCU. For serial communication the designer used MAX232 which converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. Lastly, the designer placed an inverter to the DTR pin of the DB9 male in which the inverted signal is then attach to the MCLR pin of the said MCU; this causes to reset the PIC16F877A if the DTR pin changes from low to high. The designers do this to have a software-reset because in C# DTR pin of RS-232 serial port can be altered programmatically.

Software Development

This section provides the overview of the steps that the designers undergo in developing the software of the said design. Software Development discusses the program flow and the data flow of the said design. Major compositions of this design are discussed in terms of flow chart and data flow diagram to see how the software handles the data and to see what processes the data should undergo.

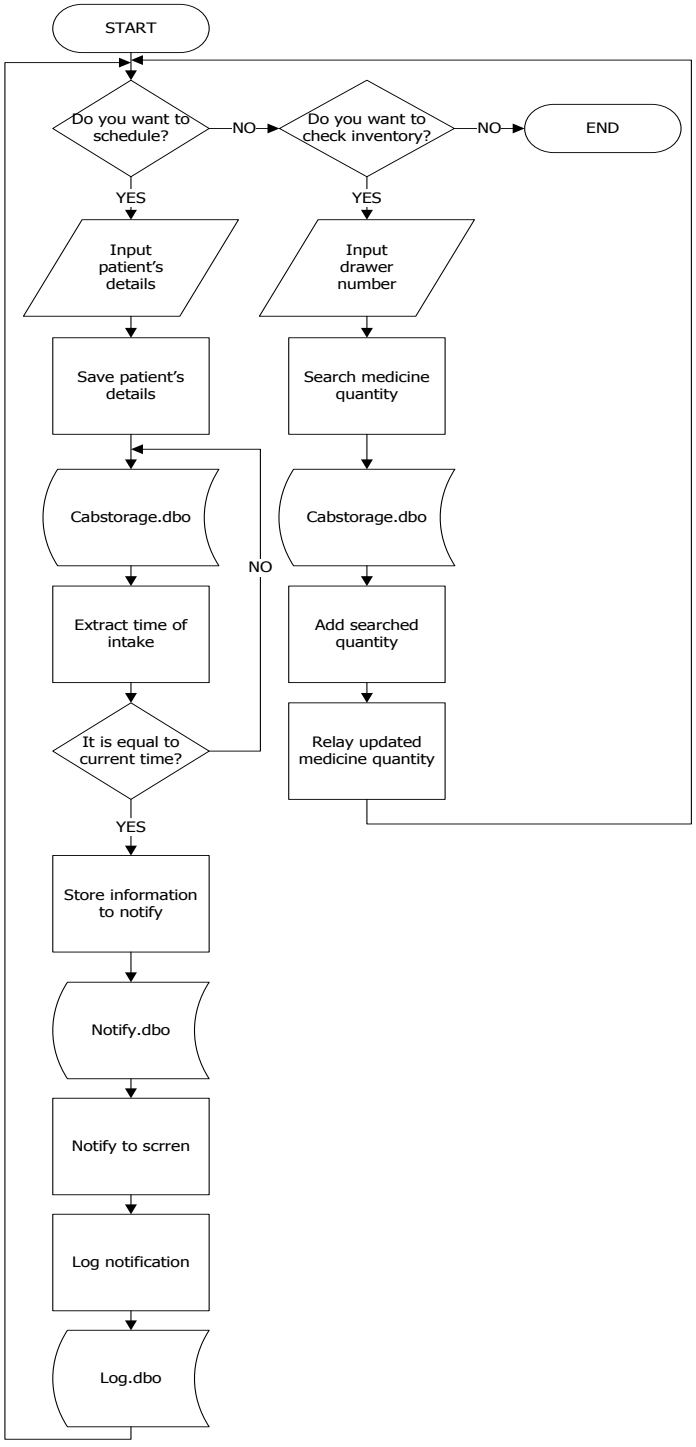


Figure 3.4 Program Flow for the Software Application

Figure 3.4 shows the program flow for the software application. It only covers the two important feature of the design for simplicity of the presentation.

The first feature is the scheduling, once the user wants to schedule he/she must enter the necessary informations on the application then the application store those informations in a database table named as “cabstorage”. If the application is running, it checks the time of intake of each medicine stored in the “cabstorage” table. Once the time of intake is equal to current time, the application will place all the necessary information to a database table named as “notify” then it will relay those information to the user. Every time that the application notifies the user, it also log those informations that was relayed to the user. Again the application will store it to another database table called “log”.

The second important feature is the inventory. The application subtracts a particular quantity to the total quantiy of medicine in a particular dawer. The subtraction is done when the notification event takes place. The user can check the inventory in which in return the application will relay the updated quantity of medicine.

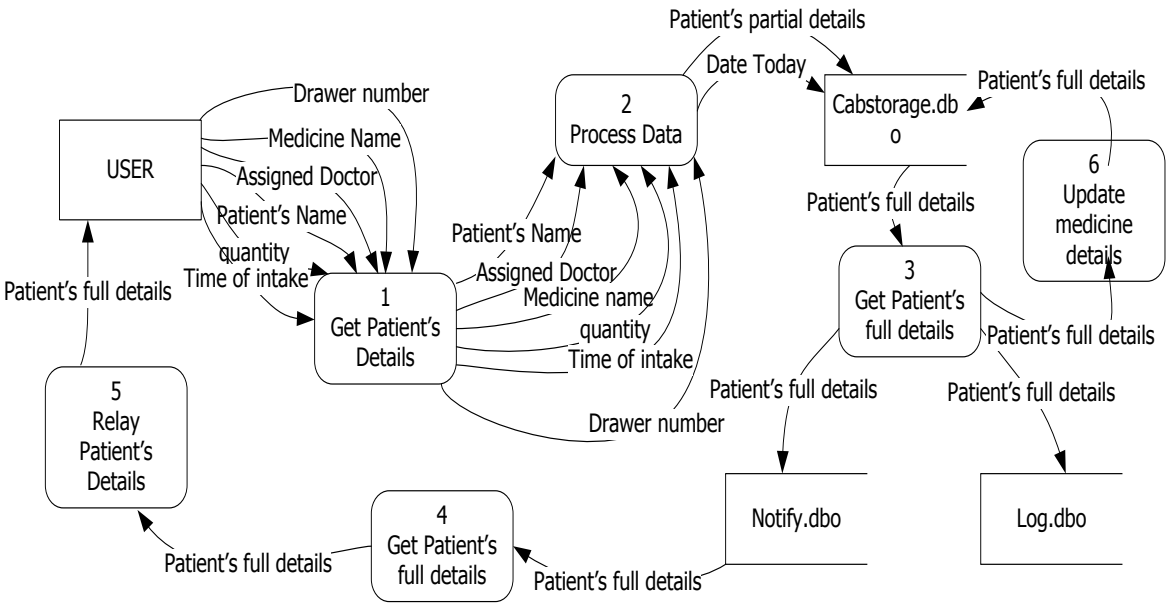


Figure 3.5 Data Flow Diagram for Scheduling

Figure 3.5 and 3.6 show the Data flow Diagram for scheduling and inventory process respectively, for the software application of this design solution. Each shape has their meanings which are the following: the circular shape stands for a process; the rectangular shape stands for storage; and the arrows stand for data that enter a process or an output of a process.

Figure 3.5 depicts the data flow for scheduling a patient's time for taking of his/her medicine/s. Initially, the user will enter the drawer number, name of patient, assigned doctor and the medicine together with the medicine's quantity and time of intake of medicine. Next; at process number 2, it will add the current date to the patient's partial details which will become an entry on the database table "cabstorage.dbo". If the current time is equal to the scheduled time the application system will get the patient's full details and store to table "notify.dbo"

and "log.dbo" (see process number 3). The "log.dbo" database table will serve as a record for all the previous notifications while the "notify.dbo" database table will serve as container for all the notifications that must be displayed for the user (see process number 4 and 5). Lastly, the "cabstorage.dbo" database table must be updated, the updating process include the subtraction to the total quantity and changing the date for the next medicine intake (see process number 6).

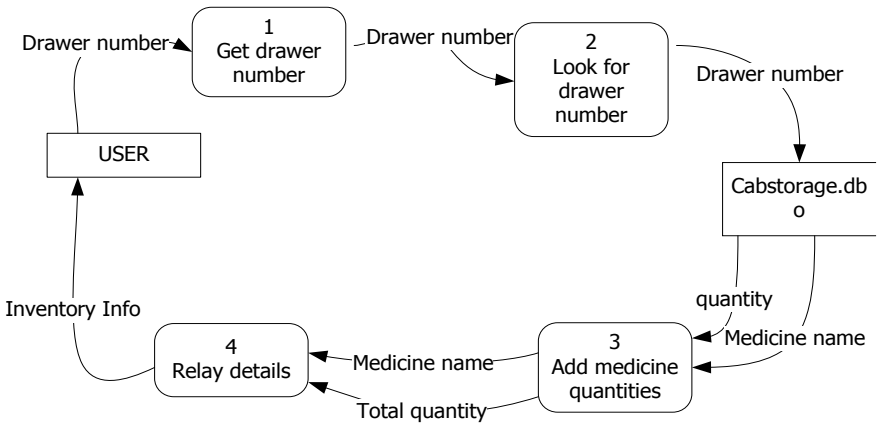


Figure 3.6 Data Flow Diagram for Inventory

Figure 3.6 depicts the Data flow for inventory process. So first the user must select a drawer number (see process number 1) after that the software application will get the drawer number assigned, and search all the scheduled entry on the said drawer, and compare if the drawer number is equal to the drawer number chosen by the user in "cabstorage.dbo" database table. After that, process number 3 will get the necessary details for getting the total number of medicines in the chosen drawer number. Lastly, process 4 will relay all the content details of the chosen drawer number to the user.

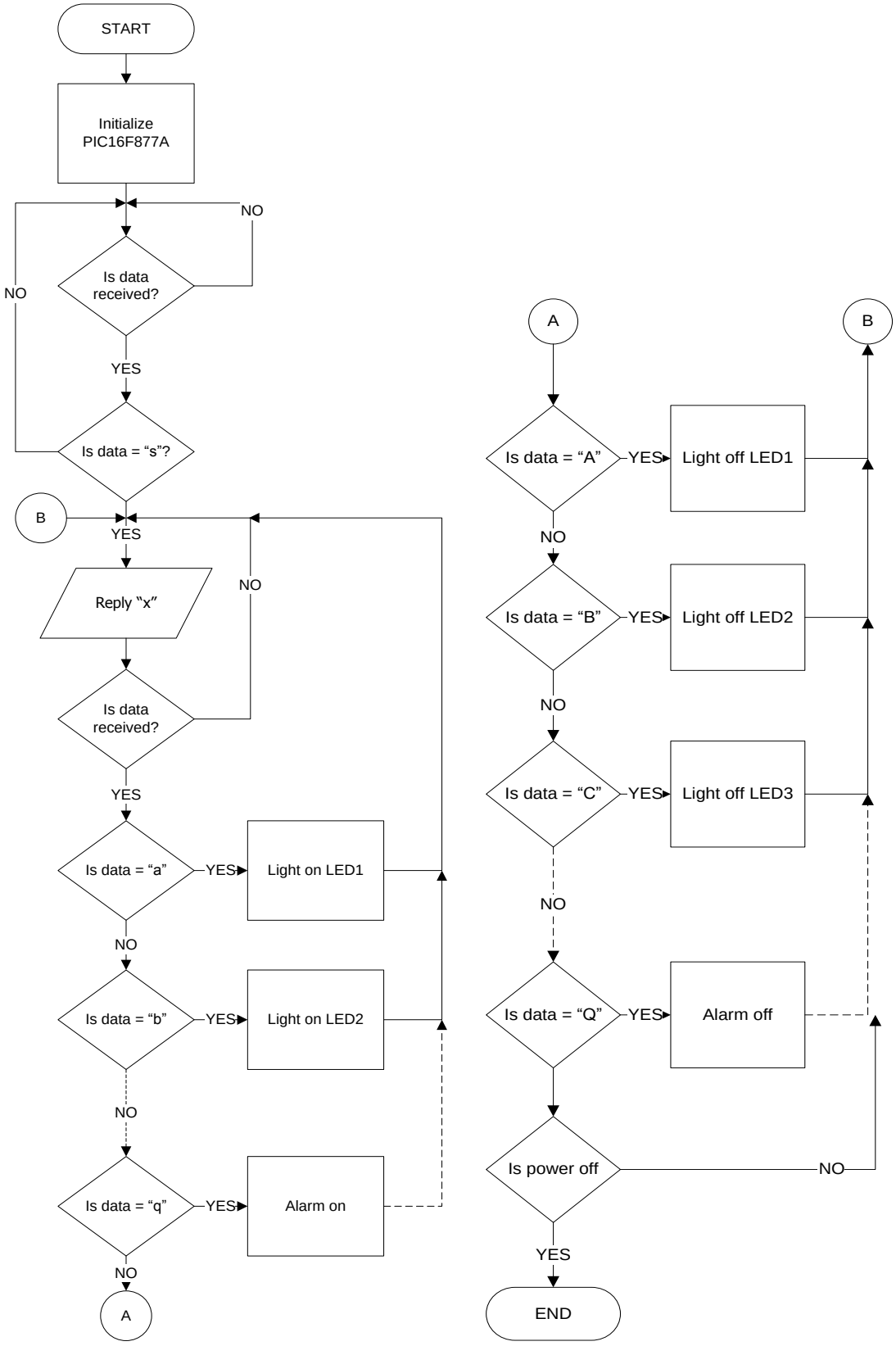


Figure 3.7 Program Flow for PIC16F877A

Figure 3.7 shows how the PIC16F877A program flows. First, PIC16F877A will initialize the ports assigned for each LEDs and Buzzer. PIC16F877A will wait for the character letter "s", the sending of this letter serve as a triggering event for the PIC16F877A to start its job; automatically PIC16F877A will send a reply letter "x".

There are specific letters assigned for each LED and buzzer. Small letters from "a" to "p" is assigned to switch on the LEDs and capital letters from "A" to "P" is assigned to turn off the LEDs. Small letter and capital letter is assigned for a specific drawer. For example, small "a" is assigned to turn on the LED in the first drawer and capital "A" is assigned to turn off the LED in the same drawer. For the second drawer letters "b" and "B" are used so on.

Small letter "q" is assigned to turn on the buzzer and small letter "Q" is assigned to off the buzzer. The buzzer will be in on state if letter "q" is sent serially, but not until "Q" is sent. The software application is the one who responsible for how long will the buzzer will be at its on state.

Therefore, one of the tasks of the software application is to send and receive data coming from/to the PIC16F877A. With that in effect the software application controls the sixteen LED and the buzzer, while the microcontroller serve as the driver for the sixteen LED and the buzzer.

Prototype Development

This section provides the overview of the steps that the designers undergo in developing the prototype of the said design.

The designers used the following software application in the prototype development Visual Studio 2010 Professional, MikroC v3.2, and Proteus 7 Professional. First, the designers gather information about the design of the circuit. Second, the designers construct the circuit and simulate it using a software simulator; in this case, the designers used Proteus 7 Professional. Third, the designers made a version of the circuit in breadboard. Fourth, the designers programmed the PIC16F877A microcontroller using MikroC v3.2, and then the designers tested it. Fifth, the designers started to code the software application using Visual Studio 2010; then the designers integrated the circuit with the software application and tested it to identify errors. And lastly, after the correction of errors the designers assembled the circuit in Printed Circuit Board or PCB then tested the whole design system with respect with the objectives presented in chapter one.

Chapter 4

TESTING, PRESENTATION, AND INTERPRETATION OF DATA

This chapter discusses the various tests conducted in relation to the stated objectives in Chapter 1 to determine the functionality and reliability of the created prototype.

Before the test, the researchers planned to have five trials on each drawer as well as with the buzzer. Using the UART terminal in Mikro C the researchers send an ASCII character to the COM port of the computer then eventually the microcontroller will interpret that character to a command that will turn on or turn off a particular LED on the drawer or will alarm the buzzer.

Table 4.1 shows the output action of the microcontroller to a specific data input. Table 4.1 serves as the basis for determining if a data sent was successfully interpreted by the microcontroller.

Table 4.1 Expected Action by the MCU with the corresponding input data

AFFECTED COMPONENT	DATA INPUT	STATE OF THE COMPONENT (OUTPUT)
LED 1	a	ON
LED 2	b	ON
LED 3	c	ON
LED 4	d	ON
LED 5	e	ON
LED 6	f	ON
LED 7	g	ON
LED 8	h	ON
LED 9	i	ON
LED 10	j	ON
LED 11	k	ON

Table 4.1 Expected Action by the MCU with the corresponding input data (Continuation)

AFFECTED COMPONENT	DATA INPUT	STATE OF THE COMPONENT (OUTPUT)
LED 12	l	ON
LED 13	m	ON
LED 14	n	ON
LED 15	o	ON
LED 16	p	ON
ALARM	q	ON
LED 1	A	OFF
LED 2	B	OFF
LED 3	C	OFF
LED 4	D	OFF
LED 5	E	OFF
LED 6	F	OFF
LED 7	G	OFF
LED 8	H	OFF
LED 9	I	OFF
LED 10	J	OFF
LED 11	K	OFF
LED 12	L	OFF
LED 13	M	OFF
LED 14	N	OFF
LED 15	O	OFF
LED 16	P	OFF
ALARM	Q	OFF

Table 4.2 and 4.3 show the collected data in testing the functionality of the LED and the buzzer of the created prototype, using the UART terminal in Mikro C. The test ensures that the researchers are successful in creating the hardware component of the design solution that has a light signal on each drawer and an alarm. With that, the researchers assumed that every command signal that was sending though serial communication has a corresponding action by the microcontroller unit.

Table 4.2 Test for LED Responses on their Respective Drawers

Drawers	LED	Trial 1 (ON)	Trial 2 (OFF)	Trial 3 (ON)	Trial 4 (OFF)	Trial 5 (ON)	Percentage of Success (%)
1	1	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
2	2	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
3	3	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
4	4	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
5	5	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
6	6	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
7	7	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
8	8	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
9	9	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
10	10	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
11	11	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
12	12	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
13	13	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
14	14	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
15	15	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
16	16	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100

Table 4.3 Test for Buzzer Response

Buzzer	Trial 1 (ON)	Trial 2 (OFF)	Trial 3 (ON)	Trial 4 (OFF)	Trial 5 (ON)	Percentage of Success (%)
Alarm	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100

The percentage of success was computed based on a formula that can be express as

$$\frac{Total\ number\ of\ success\ trial}{Total\ number\ of\ trials\ performed}\ x\ 100\% \qquad (3.1)$$

As clearly seen in table 4.2 and 4.3, the medicine cabinet was successfully responsded to every data input.

But during the design process another problem arises which lead to the question about the response of the microcontroller unit to the command send by the C# program. With this question, the designers choose test inputs for the

application software. Since the main function of this solution is to schedule a patient's time of intake, the designers constructed table 4.4.

Table 4.4 Test Inputs for the Software Application.

	Trial 1	**Trial 2**	**Trial 3**	**Trial 4**
Software Application Inputs	12 am	7:30 am	12 pm	7:30 pm

Table 4.4 represents the test inputs for the software application. The time in trial 1 and trial 3 are the two extreme input values because the software application converts the time input from 12 hour format to 24 hour format; in converting a time from 12 hour to 24 hour format it is known that adding 12 to the hour part of time is necessary for those time after noon, and those time before noon remain the same. But if the time is 12 pm it should remain the same and if the time is 12 am the hour part must be subtracted to 12, because of this diffirent process in converting 12 am and 12 pm time the designers considered these times as the two extreme values.

Now, to represent a time between those extreme values the designer chose the time in trial 2 and trial 4. The said table is necessary so that the designers would not take every time in the clock, this table serve as a representative time inputs to the software application.

Table 4.5 and 4.6 show the collected data in testing the functionality of the LED and the buzzer of the created prototype, using the trials in table 4.4.

Table 4.5 Test for LED Responses on their Respective Drawers with C# code

Drawers	LED	Trial 1	Trial 2	Trial 3	Trial 4	Percentage of Success (%)
1	1	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
2	2	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
3	3	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
4	4	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
5	5	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
6	6	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
7	7	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
8	8	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
9	9	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
10	10	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
11	11	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
12	12	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
13	13	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
14	14	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
15	15	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100
16	16	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100

Table 4.6 Test for Buzzer Response with C# code

Buzzer	Trial 1	Trial 2	Trial 3	Trial 4	Percentage of Success (%)
Alarm	SUCCESS	SUCCESS	SUCCESS	SUCCESS	100

Table 4.5 and 4.6 show the response of the light signal and the alarm with the test inputs of the software application it clearly shows that the solution responded to a particular scheduled time. Again the percentage of success was measured using equation 3.1.

Chapter 5

CONCLUSION AND RECOMMENDATION

This chapter states the conclusion of the design and the recommendation by the designers for the design's further improvements.

Conclusion

The design Solid-Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode (LED) Notifier was designed, developed, constructed and tested. The designed device was able to monitor the time intake of medicine of a patient.

The design provides a more accurate time of take of medicine. The alarm and LED notifies the nurse in charge that a patient needs to take medication through this the medication of every patient is monitored.

The design has its inventory system that will monitor what medicine should be taken out and check the quantity of available medicine. Through this inventory the medicine given to the patients will be on time. The nurse assigned will be notified by the inventory that a patient needs to take what kind of medicines and amount of medicine needed to be taken.

The tests that were conducted by the designers show the planned output of the device. The designed device will be helpful to the doctors and nurses to monitor patient's medication.

The researchers were able to conclude the following through testing: First, the solution responded to the signal that was sending using UART terminal and lastly the solution responded with the C# program at corresponding user's test inputs.

Recommendation

The following are recommended ways to improve the design:

In terms of hardware security, the medicine cabinet with multiple drawers is can be opened by anyone, thus the cabinet is not secured by illegal access of the drawers. But this problem can be address by providing a locking mechanism in each drawer.

In terms of software security, the software application only provides a simple password protection. Future study is required to provide a more secured software application.

In terms of medicine quantity monitoring, the whole system cannot detect if the quantity recorded by the software application is equal to real quantity in a drawer. This problem can be address by providing a monitoring system that can be integrated with this design project.

In terms of the medicine cabinet size, the size of the drawer and the cabinet is not limited to what is presented in this paper.

In terms of the design of user interface as well as its functionality, the design of the software application is not limited to what is presented in this paper.

In terms of compatibility of the software application to other operating system, this paper only presented a software application that is compatible for Windows operating system, but it can be extend to other operating system.

REFERENCES

Glucksman J., et al, (1968). *Programmed Medication Dispenser.*

McLaughlin J. T., (1973). *Cabinet for Dispensing Medicines at Predetermined Times.*

Hicks T. A. and Hicks B. G., (1981). *Portable Medicine Cabinet with Timer.*

Hopkins W. G., (2000). *Summarizing Data: Precision of Measurement.*

Aberle W., Hofman M., (2003). *Driving LEDs with a PIC Microcontroller.*

Kittredge R., (2003). *Concepts for Transmitting Data from a PC to a Microcontroller.*

Kuwik P., Thomas L., York M., Crump D., Livingston D. and Squire J. C., (2005). *The Smart Medical Refrigerator.*

Foo M., Chua J., Ng J., (2011). *Enhancing Medicine Adherence through Multifaceted Personalized Medicine Management.*

APPENDIX

A. Operation's Manual

System Requirement

These are the requirements necessary to operate the device. It should be a laptop or a computer with the following features:

1. Windows Operating System: Windows XP and newer versions
2. USB Port
3. Microsoft .NET Framework 4 Client Profile
4. Minimum Memory of 512MB
5. RS232 Serial port
6. USB-to-RS232 Converte (for laptops)

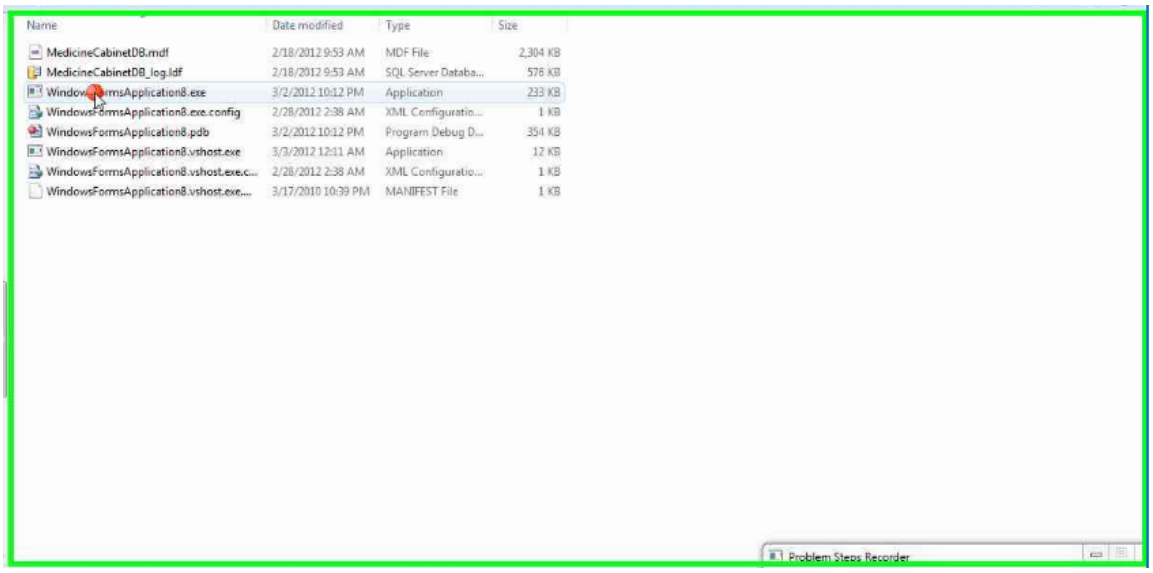
Installation Procedure

1. Connect the 9-V adapter to the medicine cabinet.
2. Connect the medicine cabinet to a RS232 serial port of a PC. If it is a laptop connect first the USB-to-RS232 converter to a USB port, then connect it to the medicine cabinet.
3. Install Microsoft .NET Framework for the software.

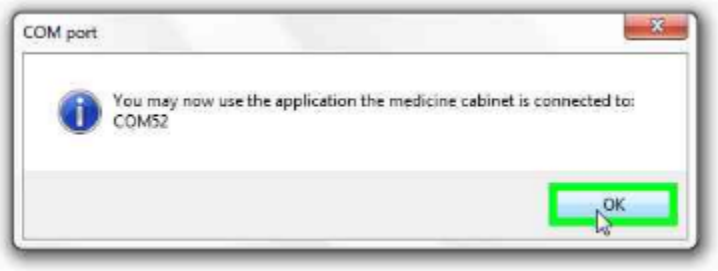
User's Manual

Opening the application:

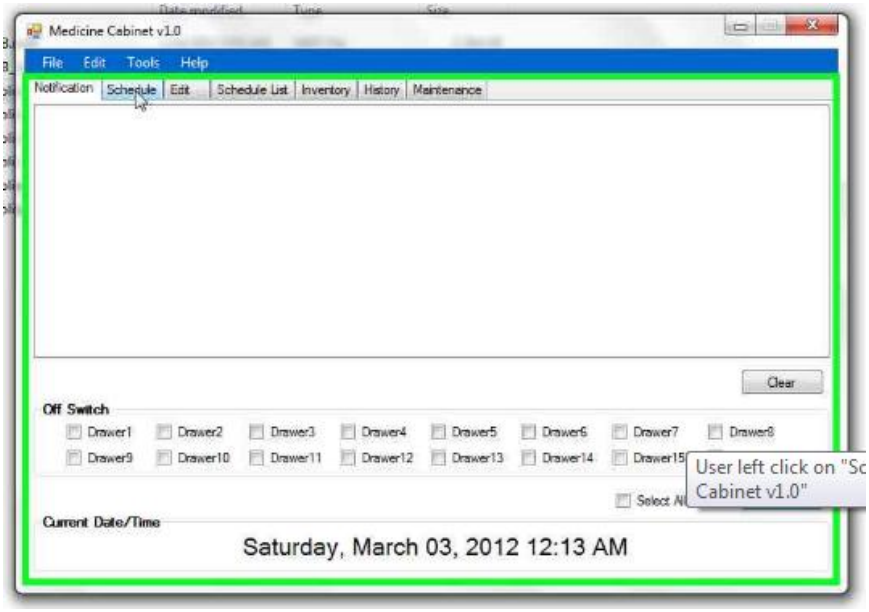
1. Double click the .exe file.



2. A message box will appear indicating that the application found the device. Click "OK".

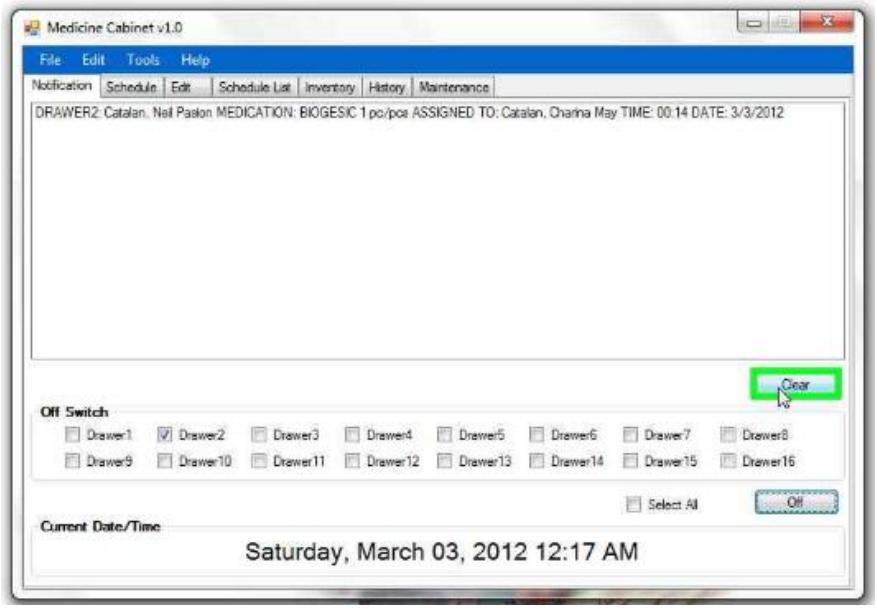


3. After clicking the "OK" button the main form will appear, take note that the first activated tab is the "Notification" tab.

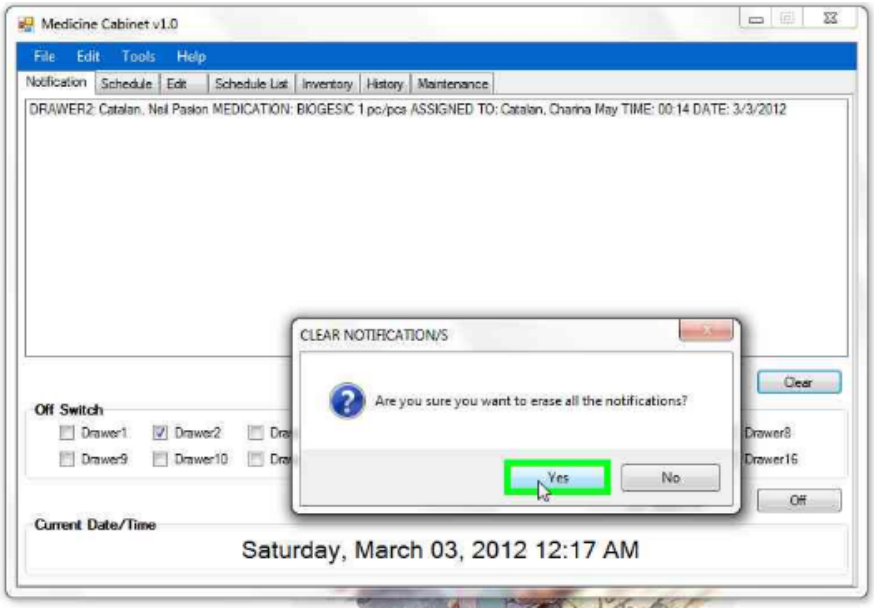

Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15
User left click on "Sc
Cabinet v1.0"
Select All
Current Date/Time
Saturday, March 03, 2012 12:13 AM

Using the notification tab:

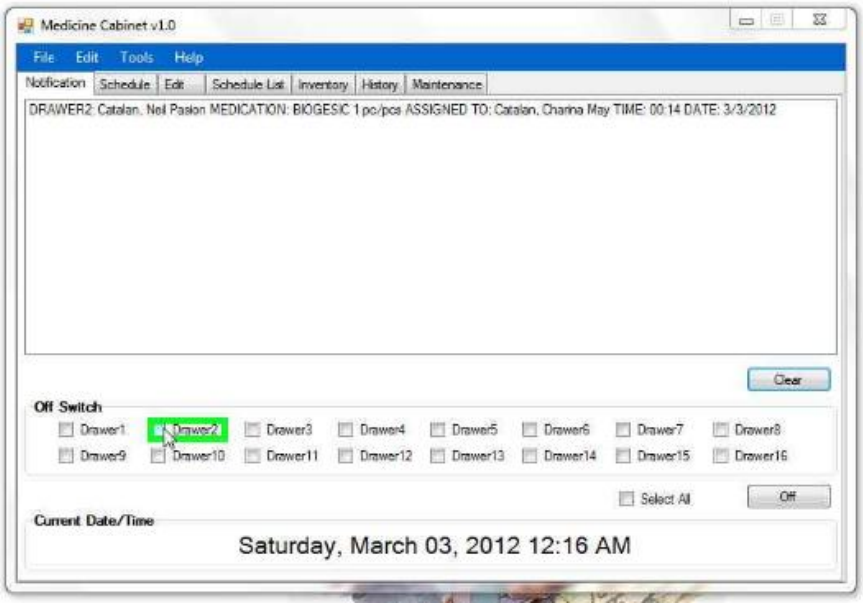
1. If a notification appeared as seen in the picture, the user can clear the area by clicking the "Clear" button.


Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
DRAWER2: Catalan, Neil Paolon MEDICATION: BIOGESIC 1 pc/pcs ASSIGNED TO: Catalan, Charina May TIME: 00:14 DATE: 3/3/2012
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:17 AM

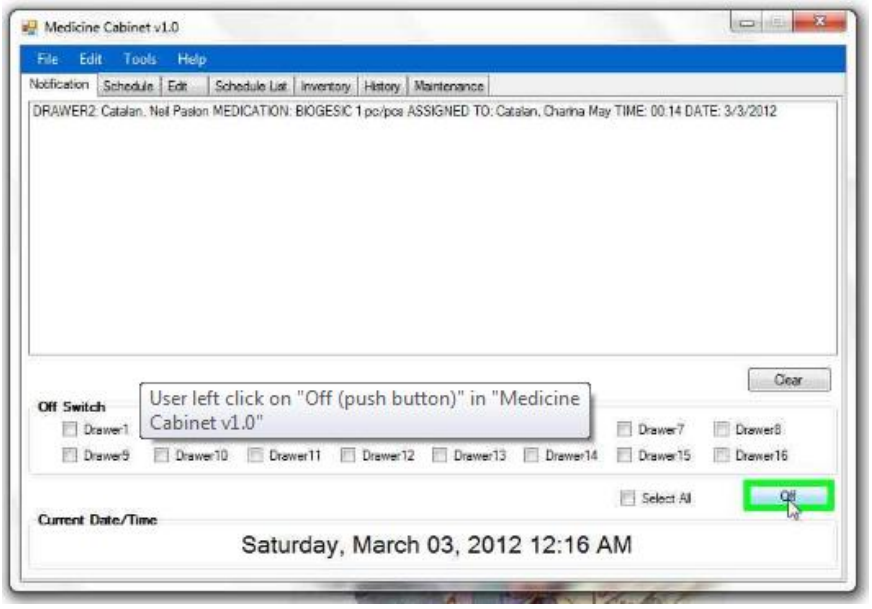
2. A message box will appear to verify the action of the user. Just click "Yes" if you want to proceed otherwise click "No".



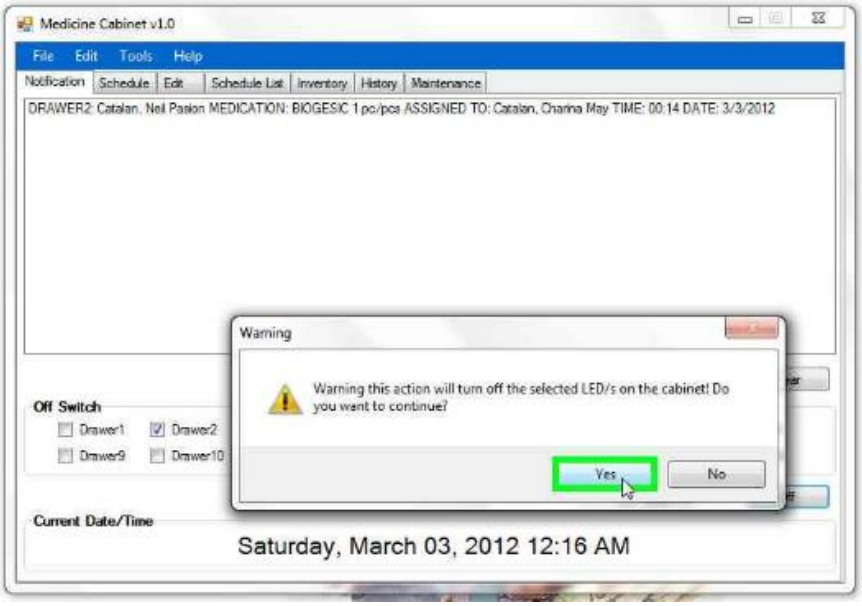
3. To turn off a particular light in a drawer select first the associated drawer number among the choices in the application. For the picture below drawer 2 is on, so we will select drawer 2.


Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
DRAWER2: Catalan, Neil Paolon MEDICATION: BIOGESIC 1 pc/pcs ASSIGNED TO: Catalan, Charina May TIME: 00:14 DATE: 3/3/2012
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:16 AM

4. Then click the "Off" button.

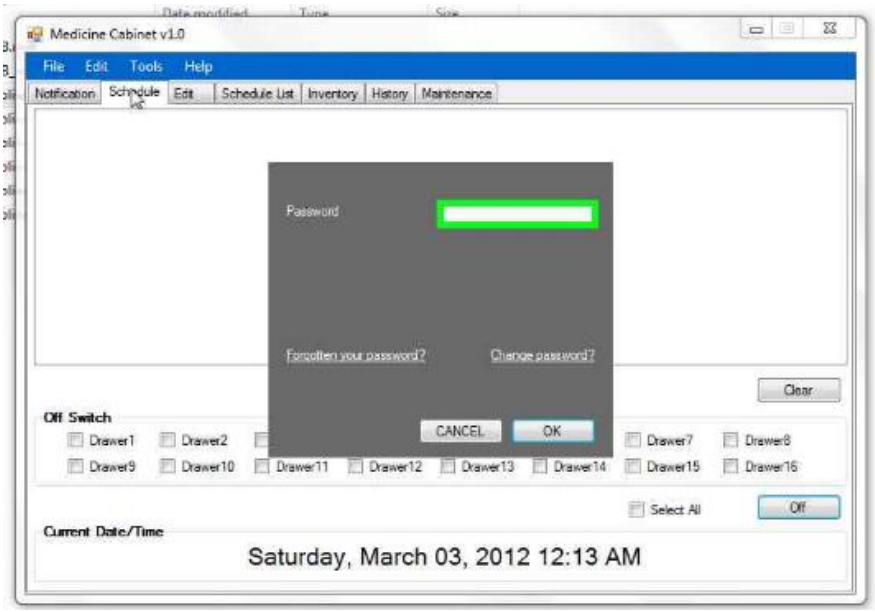

Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
DRAWER2: Catalan, Neil Paolon MEDICATION: BIOGESIC 1 pc/pcs ASSIGNED TO: Catalan, Charina May TIME: 00:14 DATE: 3/3/2012
Clear
Off Switch
User left click on "Off (push button)" in "Medicine Cabinet v1.0"
Drawer1 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:16 AM

5. A message box will appear which will verify the user action. Click "Yes" if you want to proceed otherwise click "No".

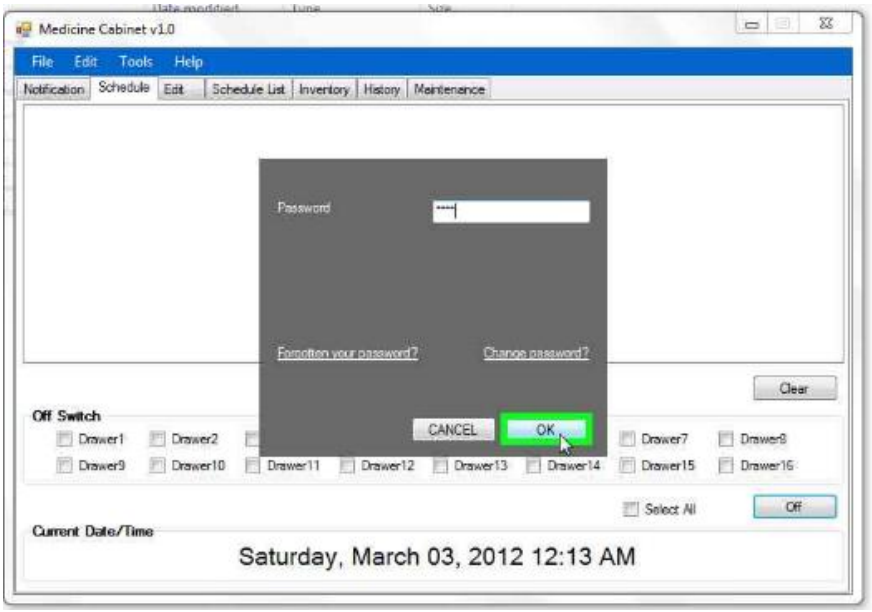

Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
DRAWER2: Catalan, Neil Paolon MEDICATION: BIOGESIC 1 pc/pcs ASSIGNED TO: Catalan, Charina May TIME: 00:14 DATE: 3/3/2012
Warning
Warning this action will turn off the selected LED/s on the cabinet! Do you want to continue?
Yes
No
Off Switch
Drawer1 Drawer2
Drawer9 Drawer10
Current Date/Time
Saturday, March 03, 2012 12:16 AM

Scheduling:

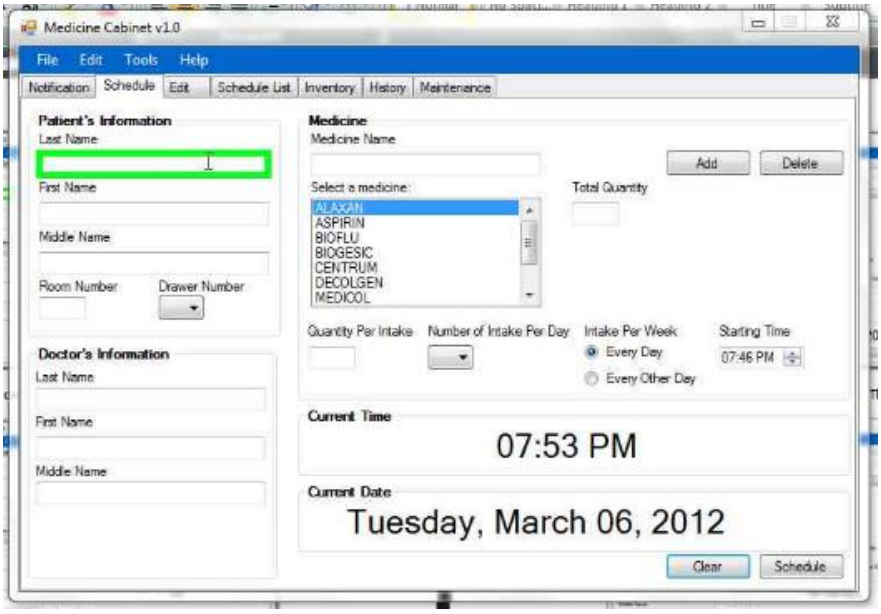
1. To schedule a medication for a patient go to "Schedule" tab. A password is required for this action.



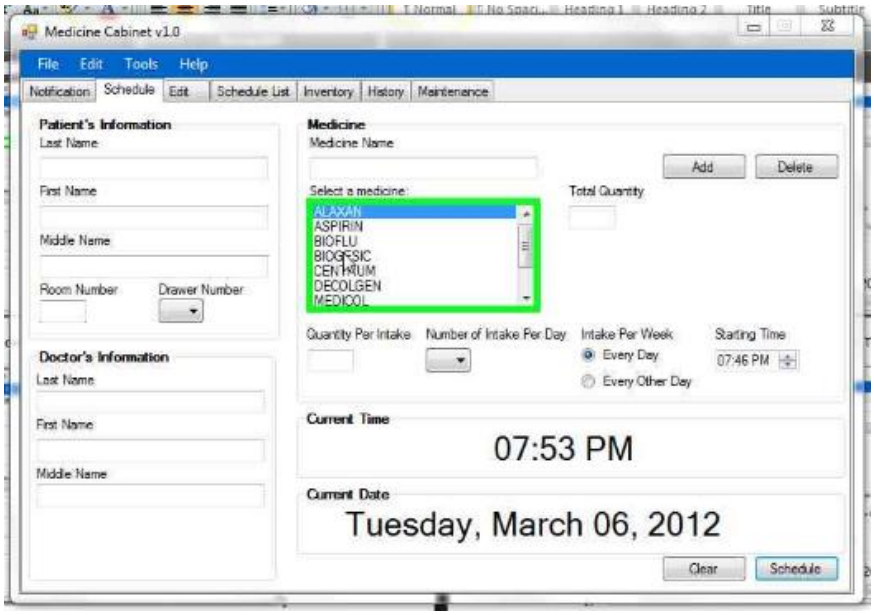
2. Enter the initial password ("neil"). The user can change the password later.



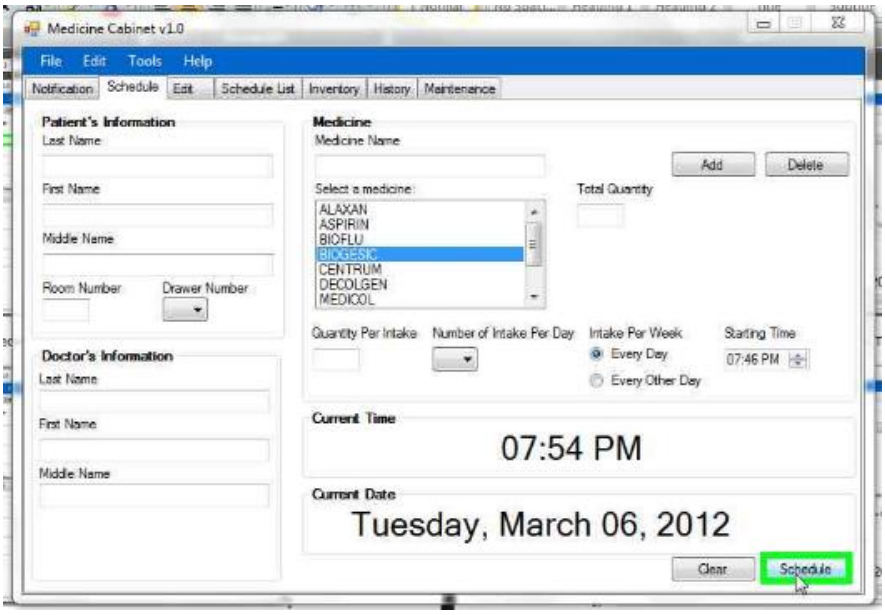
3. Type the necessary information in the form.



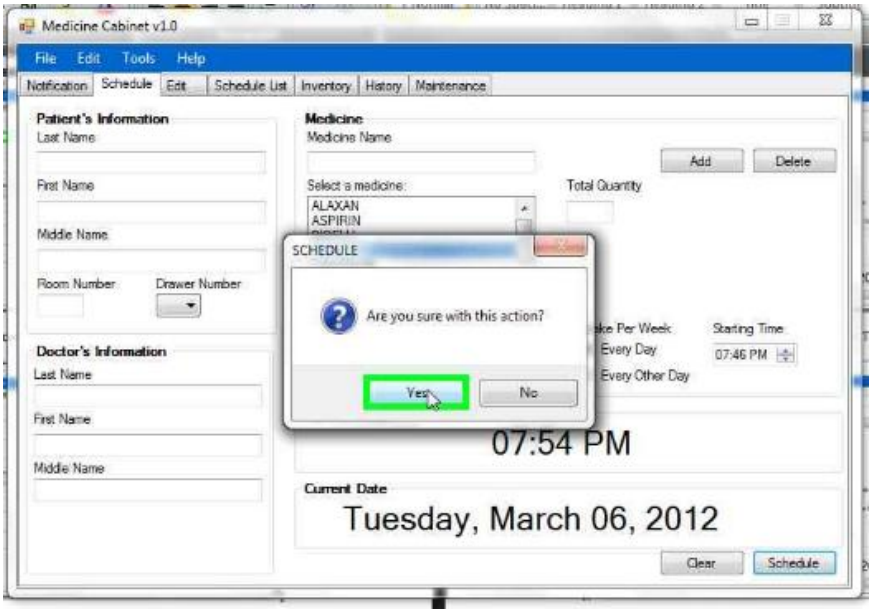
4. Select a medicine from the list box as seen in the picture. The user can add medicine by typing the name at the "Medicine Name" then click "Add". The user can also delete a medicine from the list just select first a medicine name from the list box then click "Delete".



5. Type the schedule for the selected medicine. Then click "Schedule".

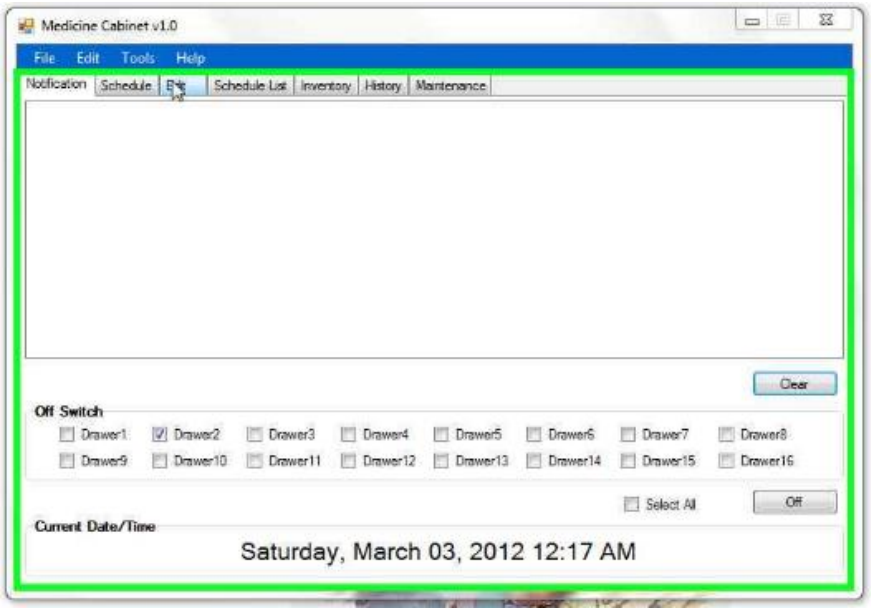


6. A message box will appear to verify the user action. Click "Yes" to proceed otherwise click "No".

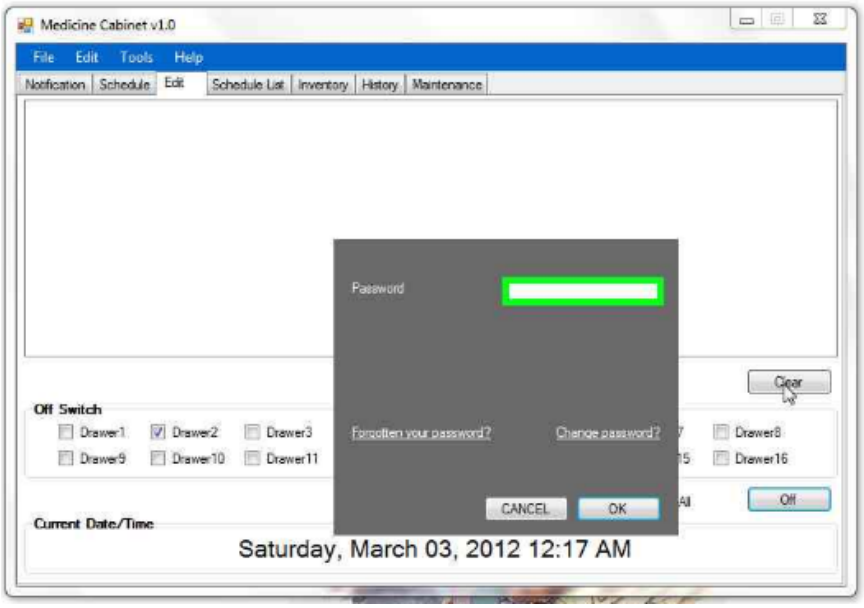


Editing a patient's entry:

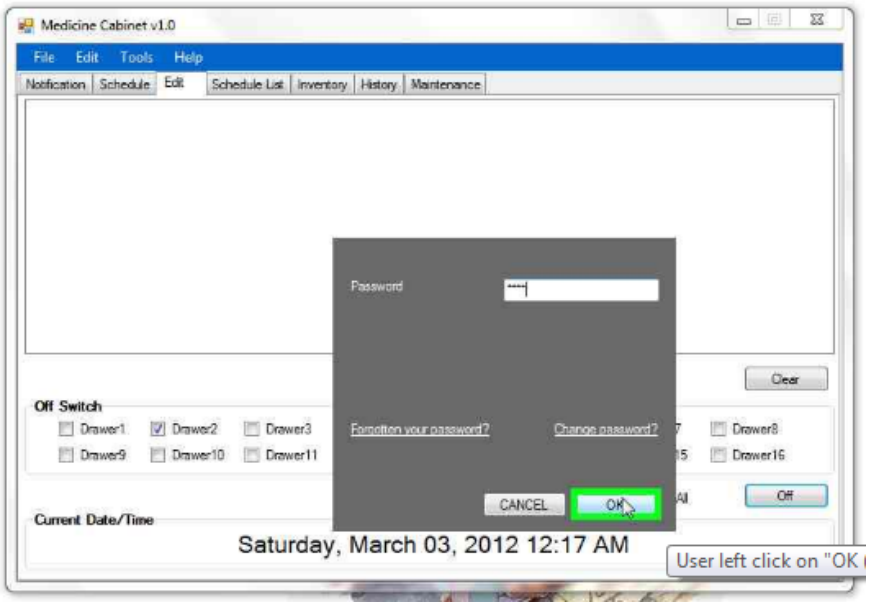
1. To edit a patient's entry go to "Edit" tab.


Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:17 AM

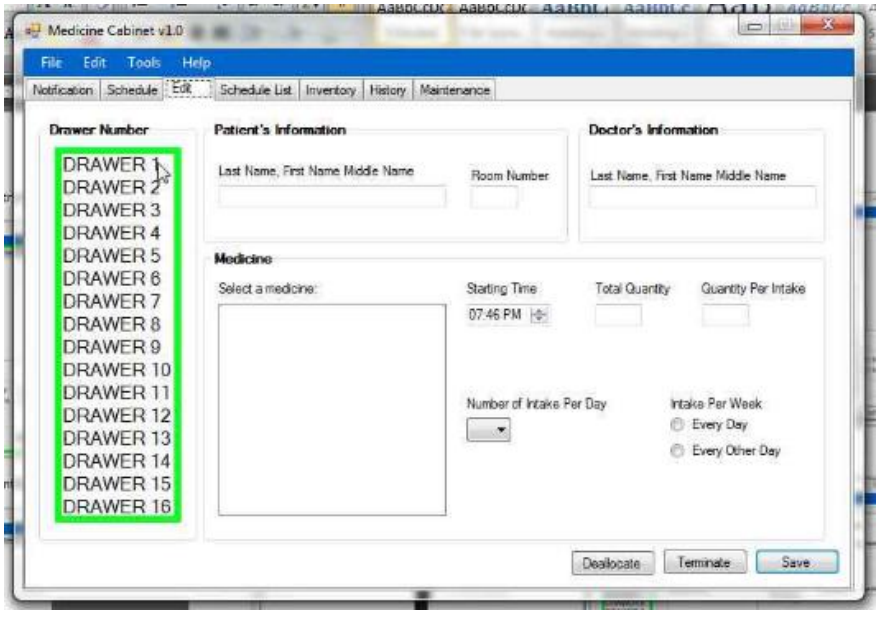
2. This action requires a password. Enter initial password ("neil"). The user can change the initial password later.



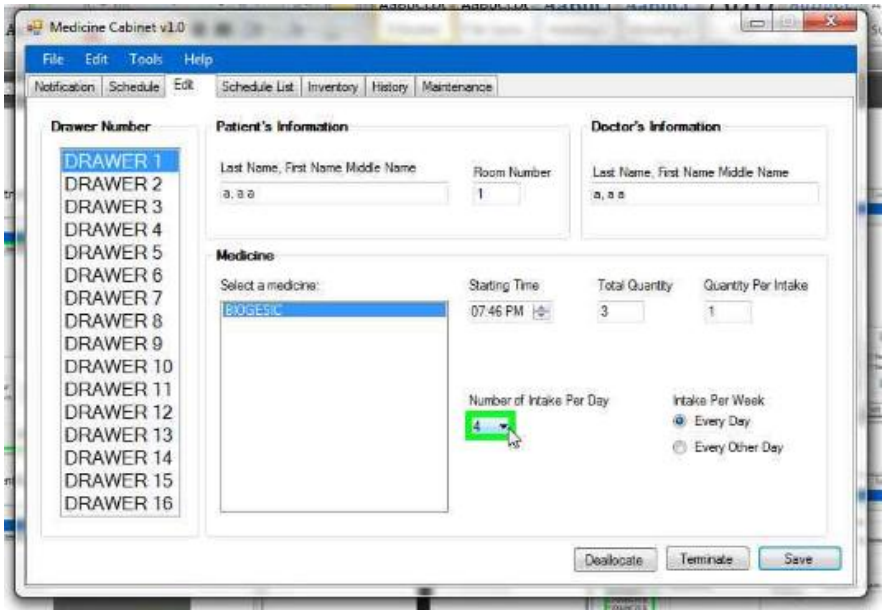
3. After entering the password click "Ok".



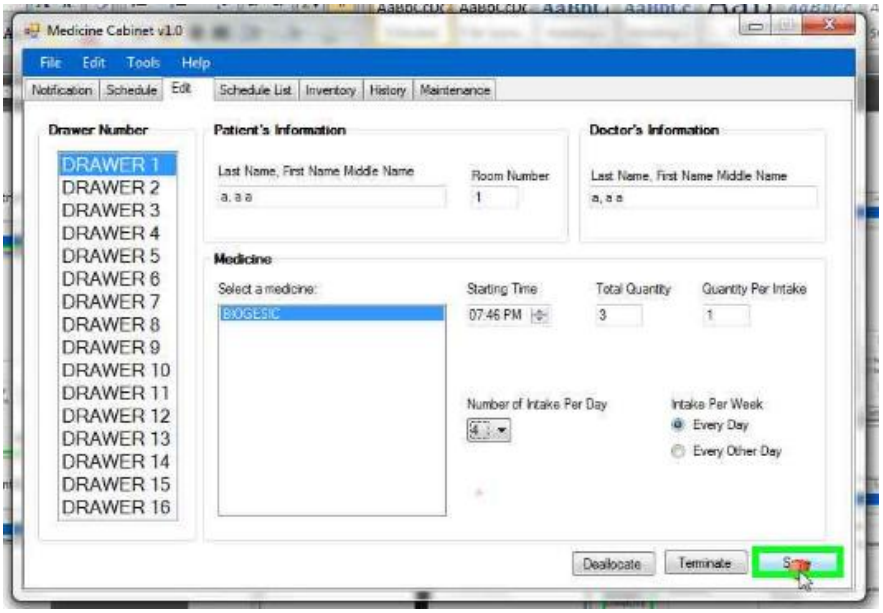
4. Select a drawer from the list box as seen in the picture.



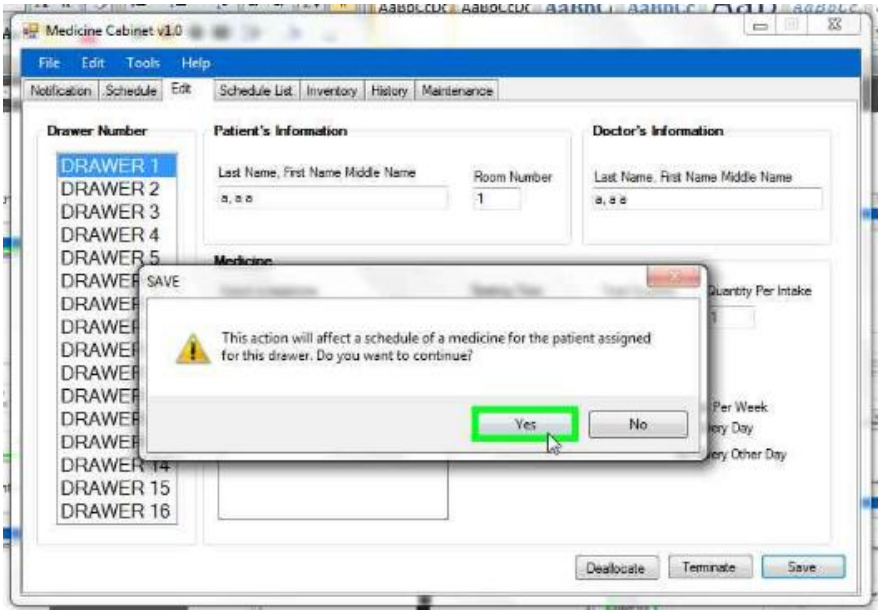
5. After selecting a drawer number it will give you the patient, doctor and the medicine informations, the user can now alter these information.



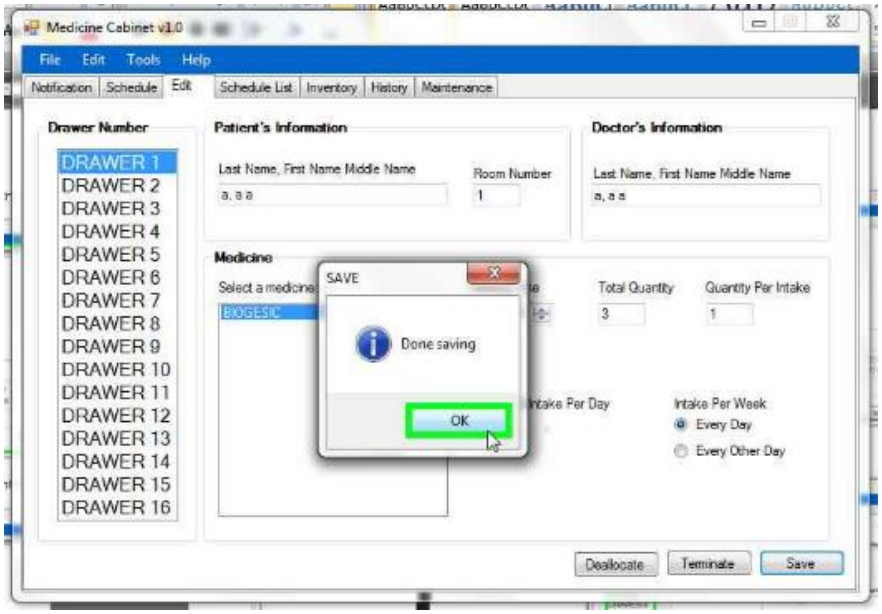
6. After altering click "Save".



7. A message box will appear, this will verify the user action click “Yes” to proceed otherwise click “No”.

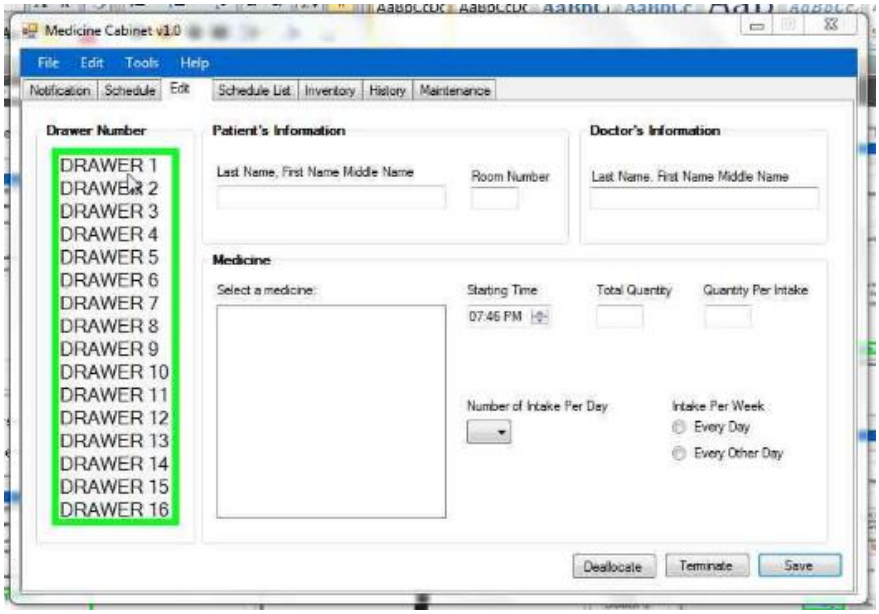


8. A message box will appear indicating that the saving of the data was done.

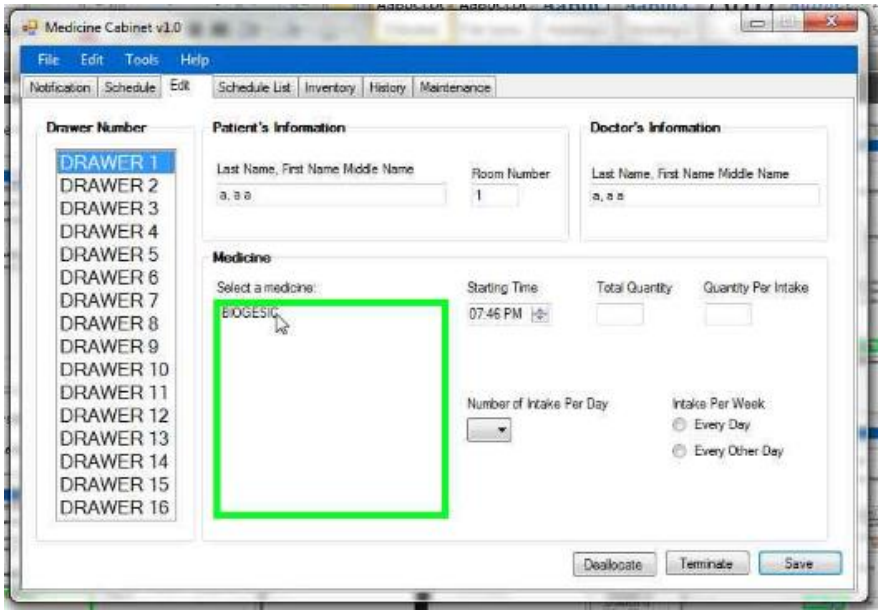


Terminating a schedule for a particular medicine in a specific drawer:

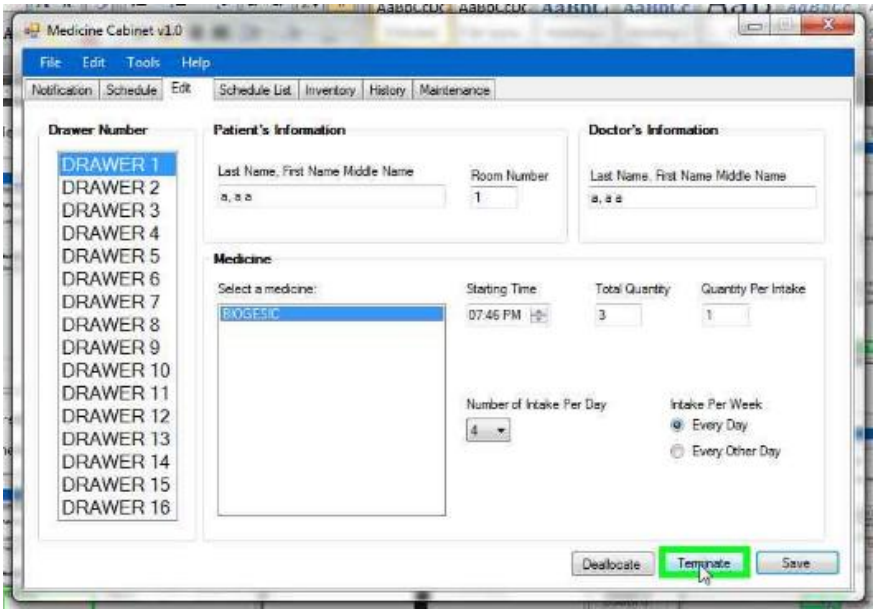
1. To terminate a schedule for a particular medicine choose first a drawer number from the list box. Take note that we are still at the "Edit" tab.



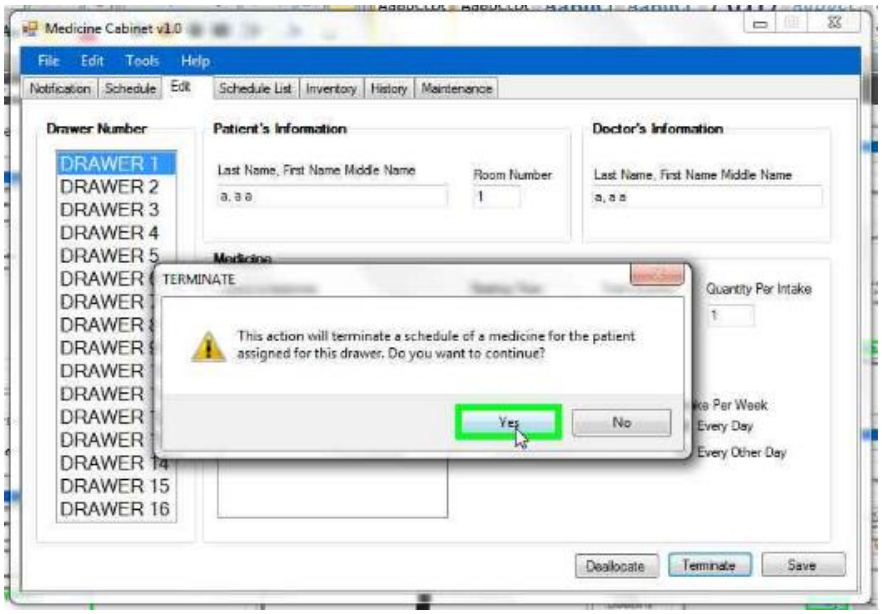
2. Then the necessary information will appear. Select a medicine from the list box as seen in the picture.



3. Click "Terminate" button.

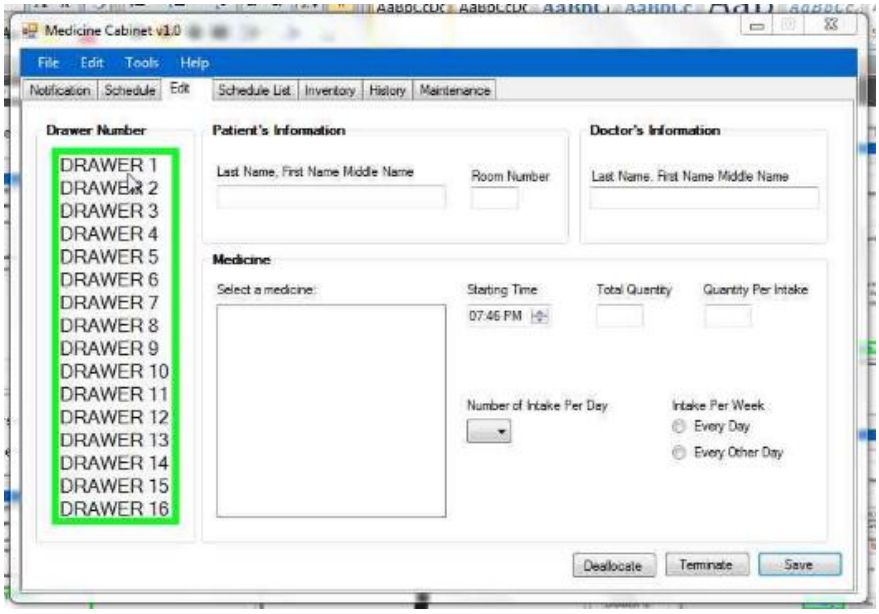


4. A message box will appear this will verify the user action. Click "Yes" to proceed; otherwise, click "No".

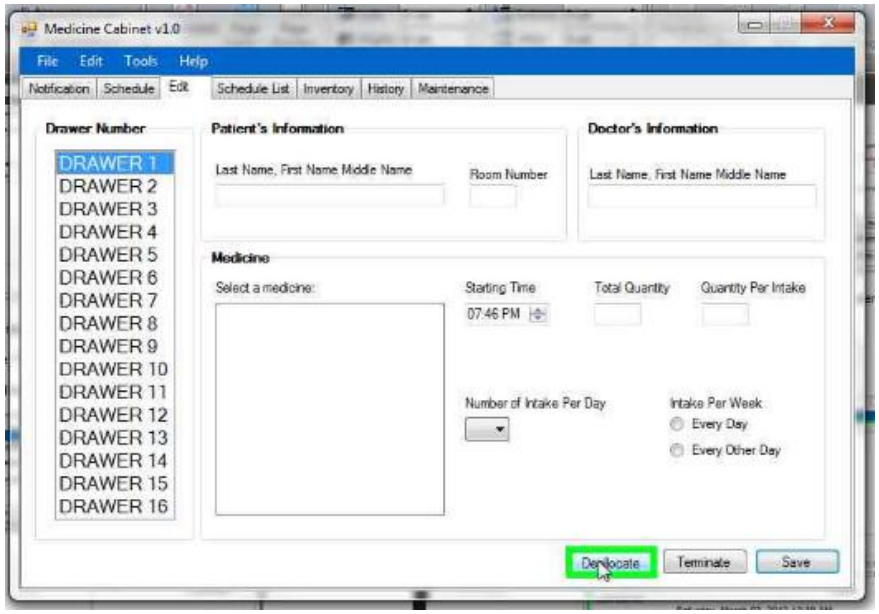


Deallocating a drawer:

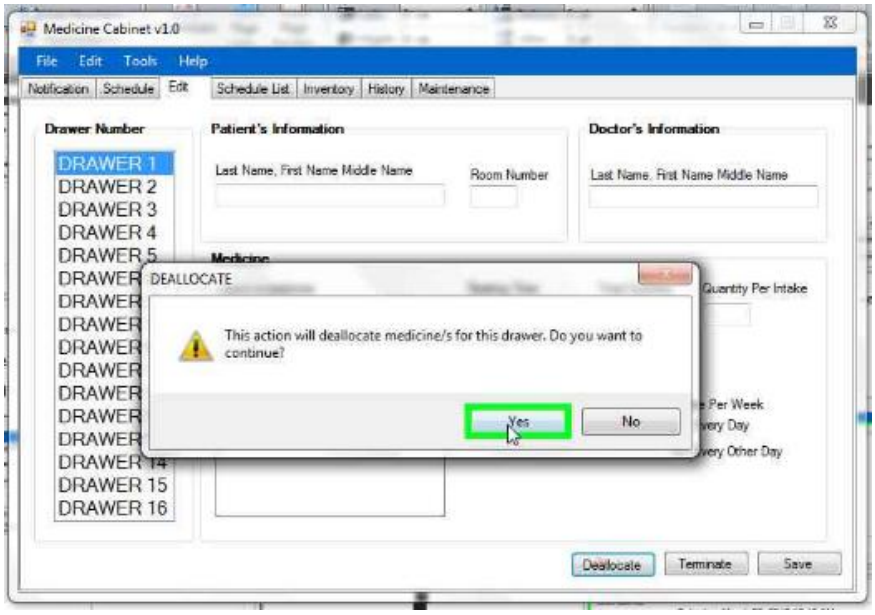
1. The user can remove the assigned patient to a drawer; this can be done by clicking the "Deallocate" button. Just be careful with this action because this will remove all the information in a drawer.



2. First, select a drawer number. Then, click "Deallocate".

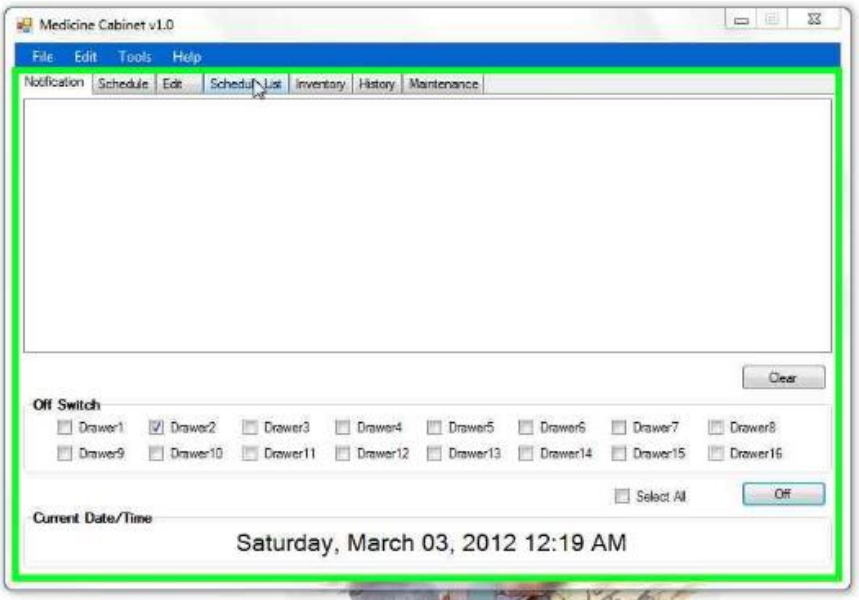


3. A message box will show, this will verify the user action. Click "Yes" to proceed otherwise click "No".

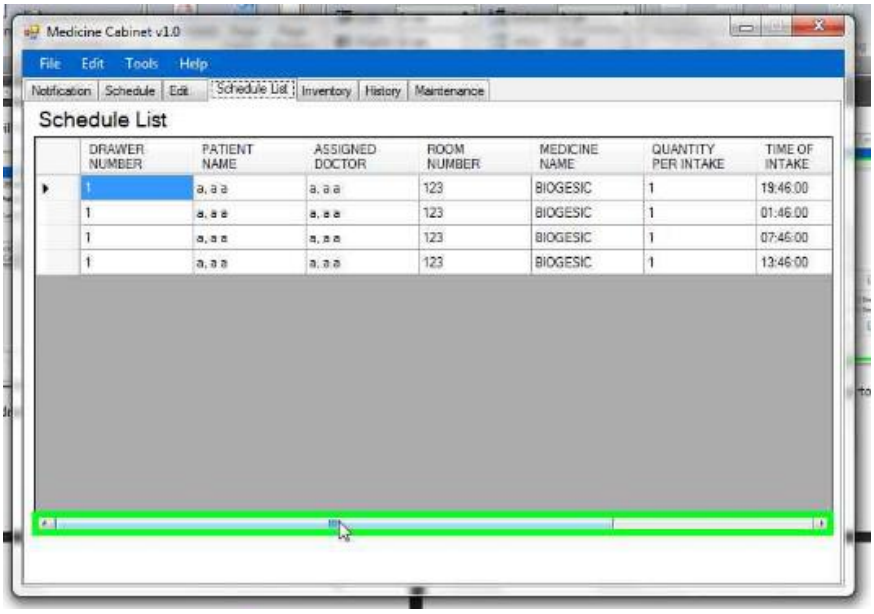


Viewing schedule list:

1. If the user wants to view the schedule list just go to "Schedule List" tab.

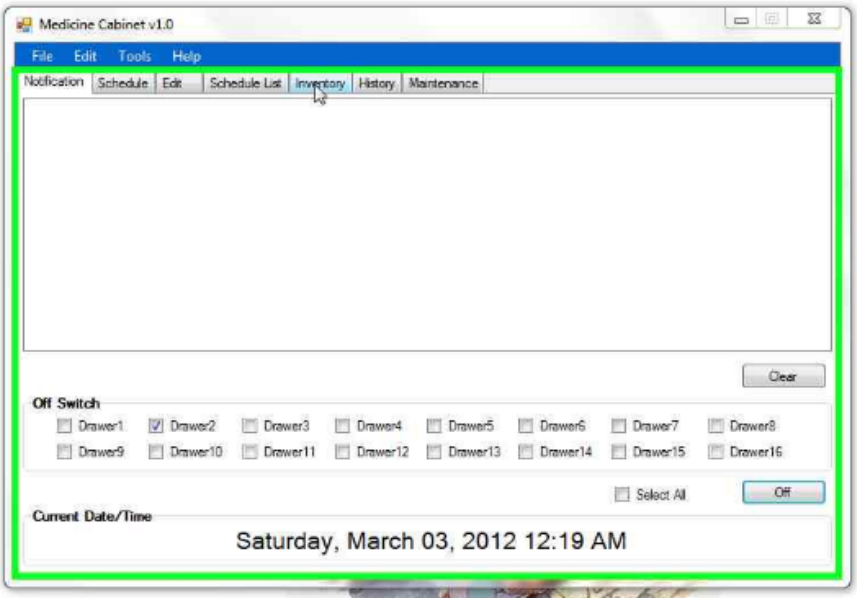

Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:19 AM

2. Then the list will show.

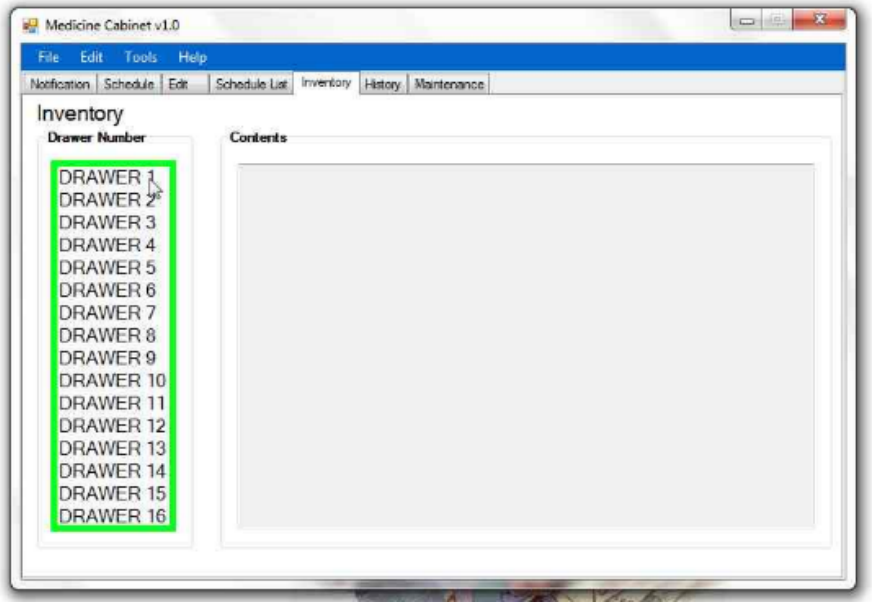

Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Schedule List
DRAWER NUMBER
PATIENT NAME
ASSIGNED DOCTOR
ROOM NUMBER
MEDICINE NAME
QUANTITY PER INTAKE
TIME OF INTAKE
1 a, a a a, a a 123 BIOGESIC 1 19:46:00
1 a, a a a, a a 123 BIOGESIC 1 01:46:00
1 a, a a a, a a 123 BIOGESIC 1 07:46:00
1 a, a a a, a a 123 BIOGESIC 1 13:46:00

Viewing inventory:

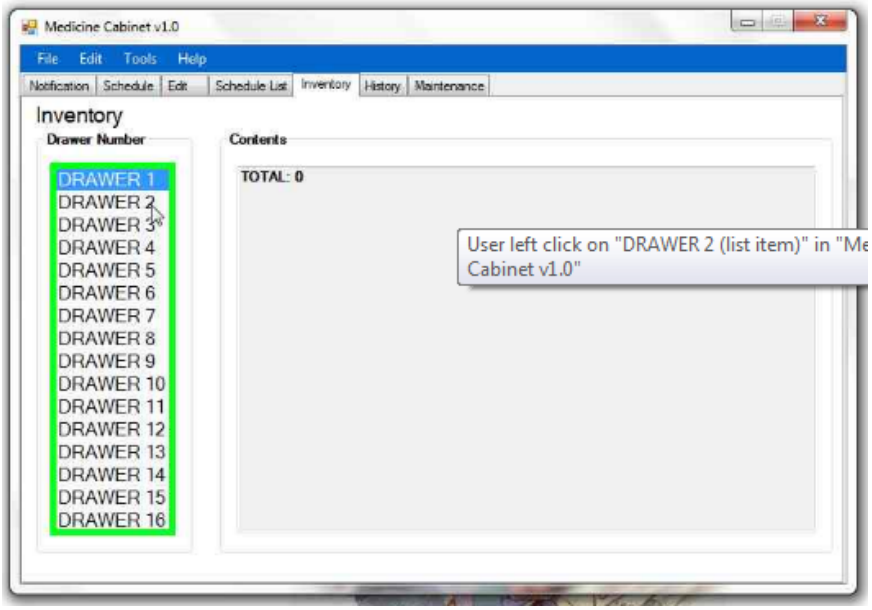
1. If the user wants to view the content of a drawer, just go to to "Inventory" tab.


Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:19 AM

2. Select a drawer number from the list box as shown in the picture.

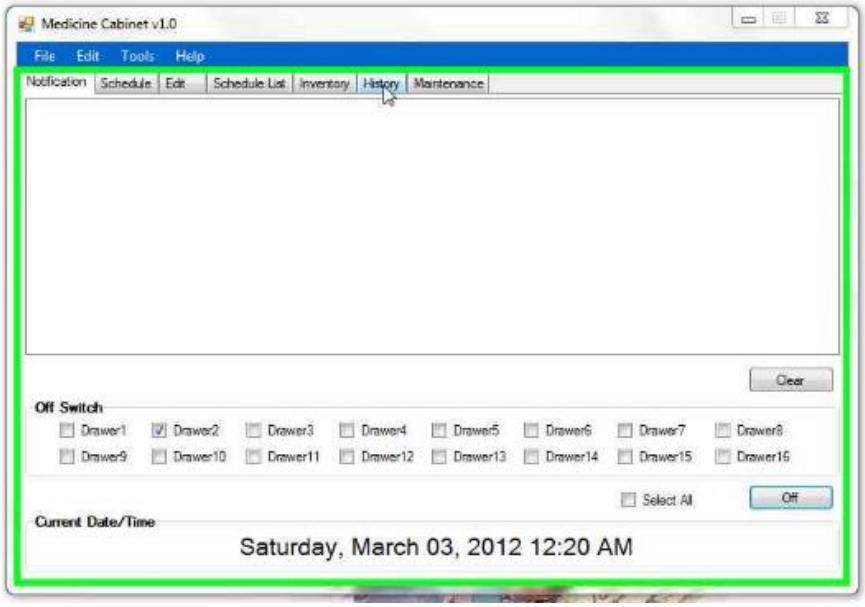


3. After selection of drawer number the contents will show at the right side of the active tab. As you can see from the picture "DRAWER 1" contains "0" (zero) medicine.

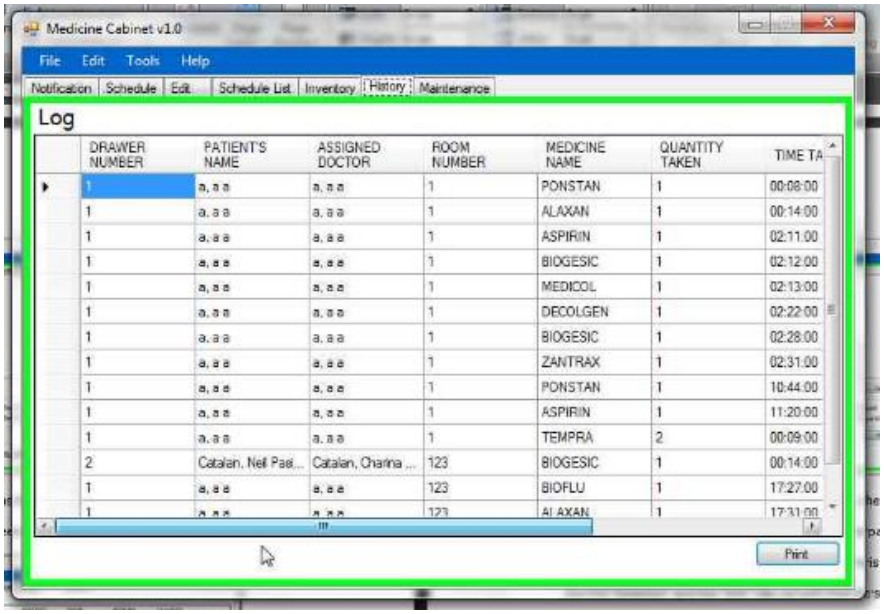


Viewing history:

1. The application has the ability to log every notification in the "Notification" tab. This can be seen at the "History" tab.

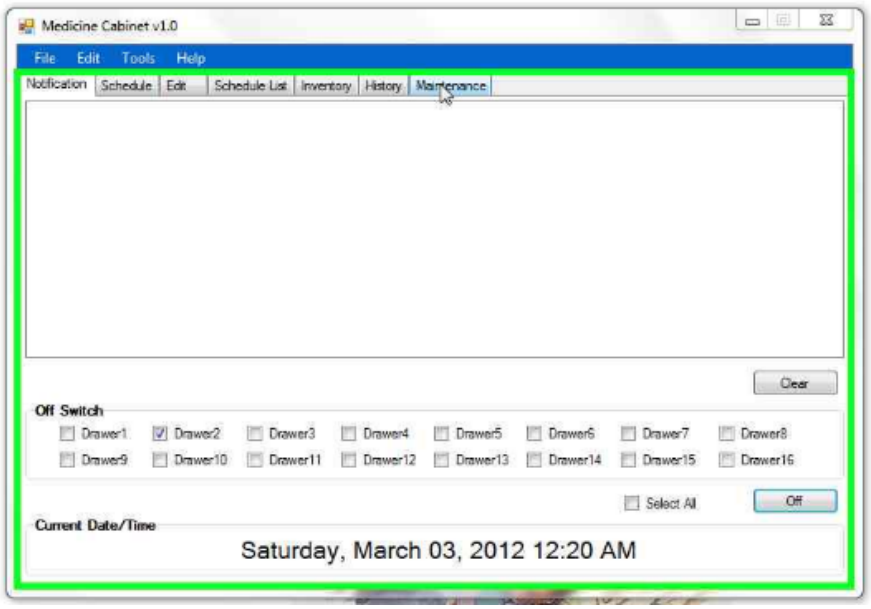

Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Clear
Off Switch
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All
Off
Current Date/Time
Saturday, March 03, 2012 12:20 AM

2. Goto to "History" tab then the log list will appear.


Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Log
DRAWER NUMBER | PATIENT'S NAME | ASSIGNED DOCTOR | ROOM NUMBER | MEDICINE NAME | QUANTITY TAKEN | TIME TA
1 | a, a a | a, a a | 1 | PONSTAN | 1 | 00:08:00
1 | a, a a | a, a a | 1 | ALAXAN | 1 | 00:14:00
1 | a, a a | a, a a | 1 | ASPIRIN | 1 | 02:11:00
1 | a, a a | a, a a | 1 | BIOGESIC | 1 | 02:12:00
1 | a, a a | a, a a | 1 | MEDICOL | 1 | 02:13:00
1 | a, a a | a, a a | 1 | DECOLGEN | 1 | 02:22:00
1 | a, a a | a, a a | 1 | BIOGESIC | 1 | 02:28:00
1 | a, a a | a, a a | 1 | ZANTRAX | 1 | 02:31:00
1 | a, a a | a, a a | 1 | PONSTAN | 1 | 10:44:00
1 | a, a a | a, a a | 1 | ASPIRIN | 1 | 11:20:00
1 | a, a a | a, a a | 1 | TEMPRA | 2 | 00:09:00
2 | Catalan, Neil Pasi... | Catalan, Charina ... | 123 | BIOGESIC | 1 | 00:14:00
1 | a, a a | a, a a | 123 | BIOFLU | 1 | 17:27:00
Print

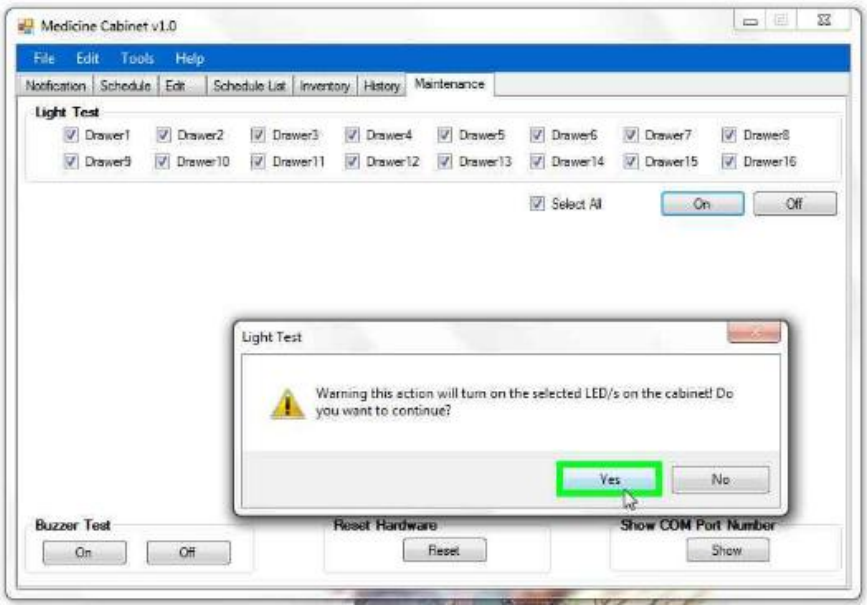
Using the maintenance tab:

1. If the user wants to check the lights in each drawer, the buzzer, and the COM port number, or the user wants to reset the hardware part of this system; the user can go to "Maintenance" tab. Take note that this is password protected like the "Schedule" and the "Edit" tab, so with that the same process applies in entering the password.

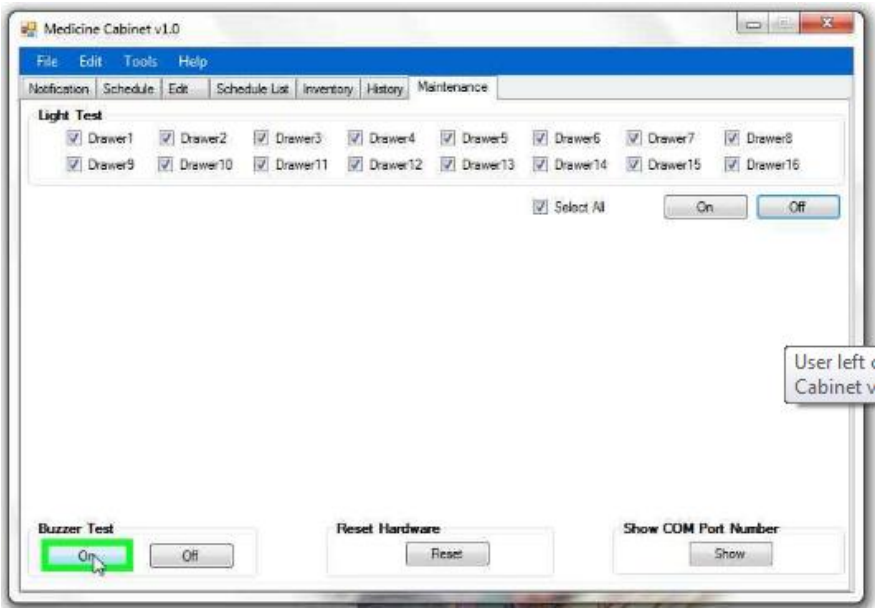


2. To test the light in each drawer, select a drawer number among the choices then click "On" this will turn on the light otherwise if the user wants to turn off the light click "Off".

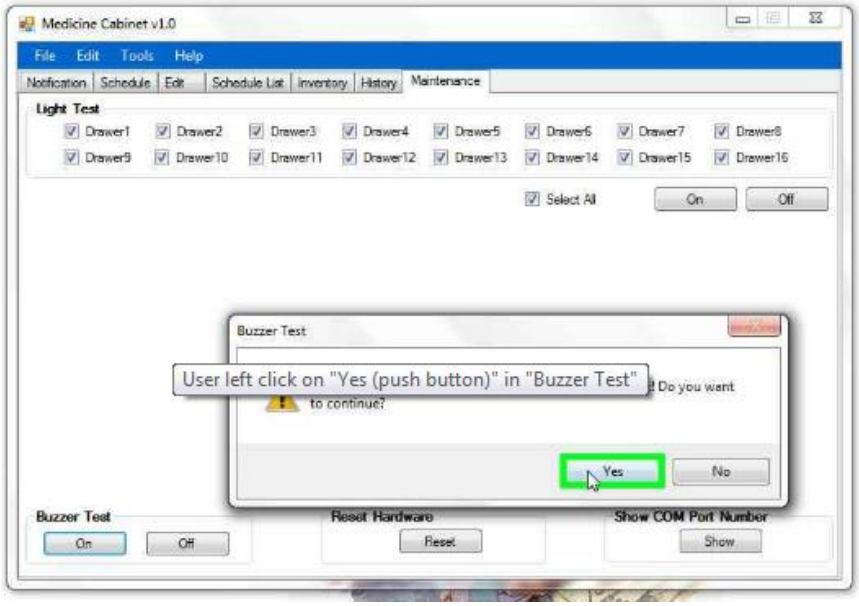
3. Again a message box will appear to verify the user action. Click "Yes" to proceed otherwise click "No".



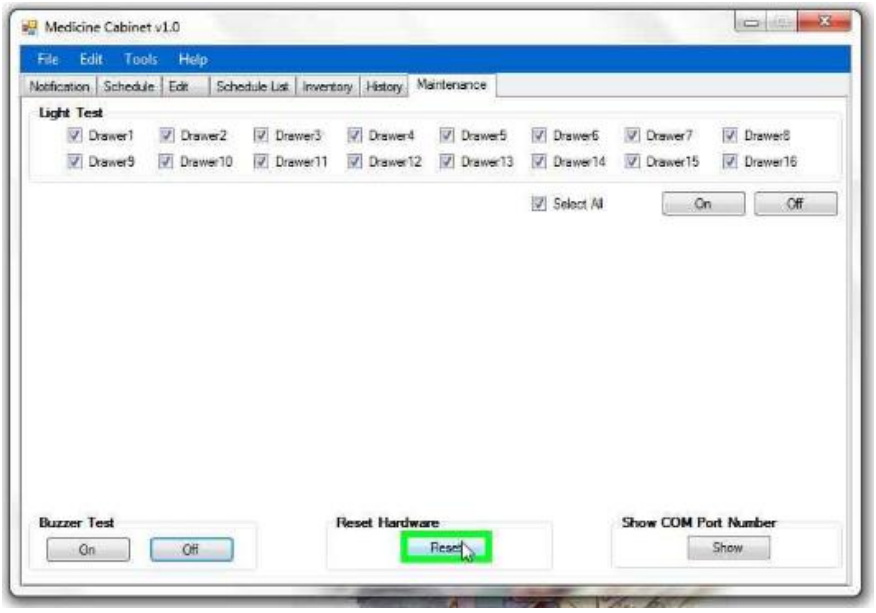
4. To test the buzzer just click “On” as shown in the picture if the user wants to turn off the buzzer click “Off”.



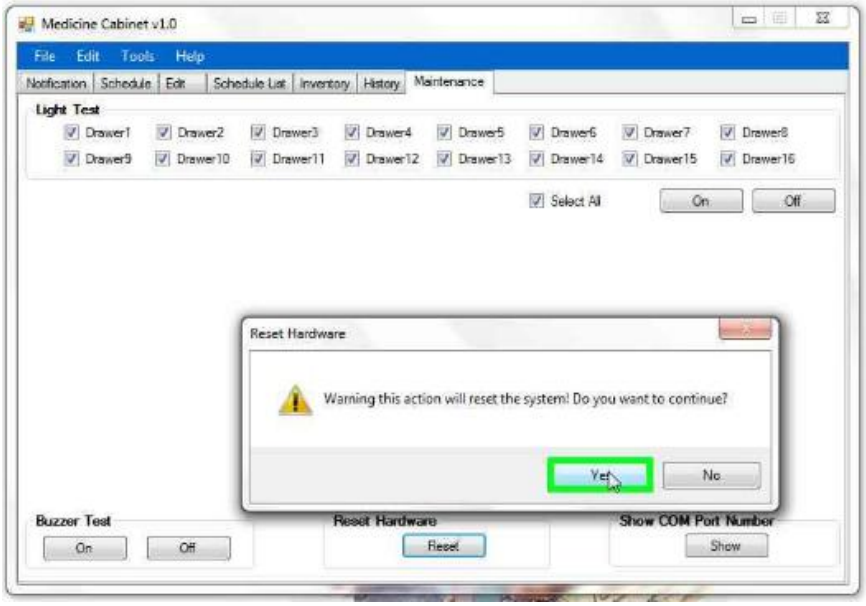
5. Again a message box will appear to verify user action. Click “Yes” to proceed.


Medicine Cabinet v1.0
File Edit Tools Help
Notification Schedule Edit Schedule List Inventory History Maintenance
Light Test
Drawer1 Drawer2 Drawer3 Drawer4 Drawer5 Drawer6 Drawer7 Drawer8
Drawer9 Drawer10 Drawer11 Drawer12 Drawer13 Drawer14 Drawer15 Drawer16
Select All On Off
Buzzer Test
User left click on "Yes (push button)" in "Buzzer Test"
Do you want to continue?
Yes No
Buzzer Test
On Off
Reset Hardware
Reset
Show COM Port Number
Show

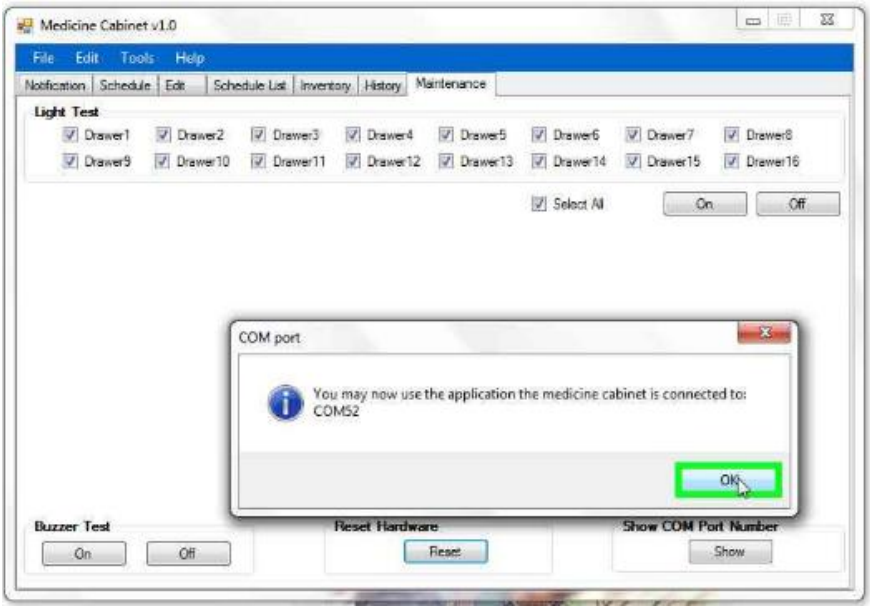
6. If the user wants to reset the hardware just click the "Reset" button. Be careful in doing this because this will reinitialize the device.



7. A message box will appear. Click "Yes" if you want o proceed.

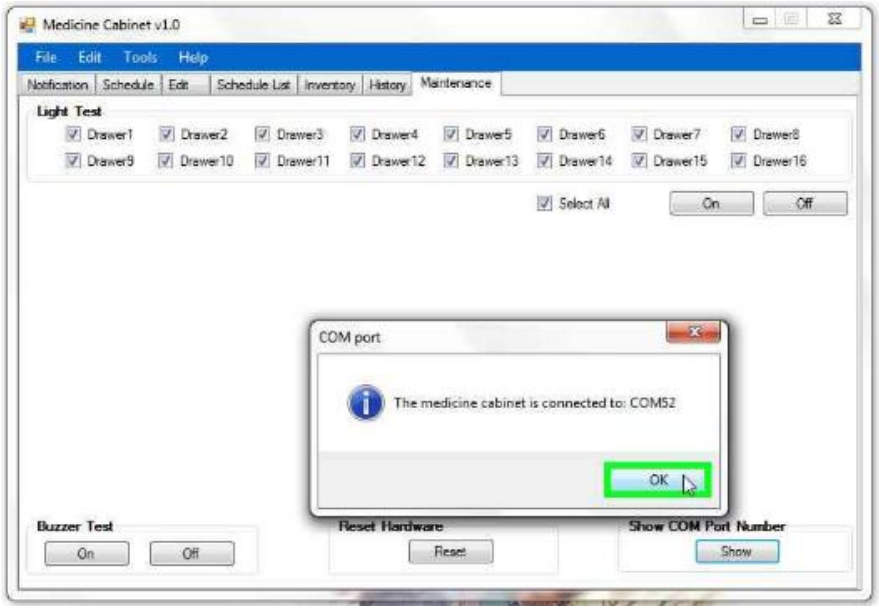


8. If the reset was succesfull a message box will appear indicating that the application found the device.



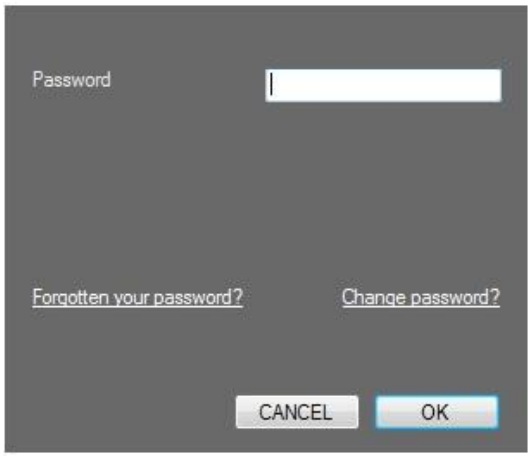
9. If the user wants to know the port used by the device click "Show" button.

10. A message box will appear indicating the port number.



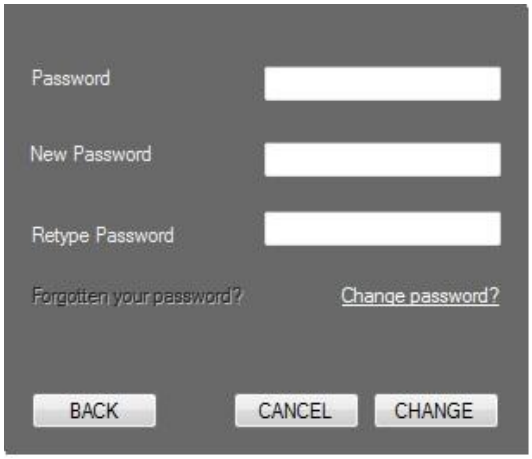
Entering password:

1. Below is the prompt for entering the password. Just type the password in the text box then click "OK". If the user forgot the password click the "Forgotten your password?" link or if the user wants to change the password click the "Change password?" link.



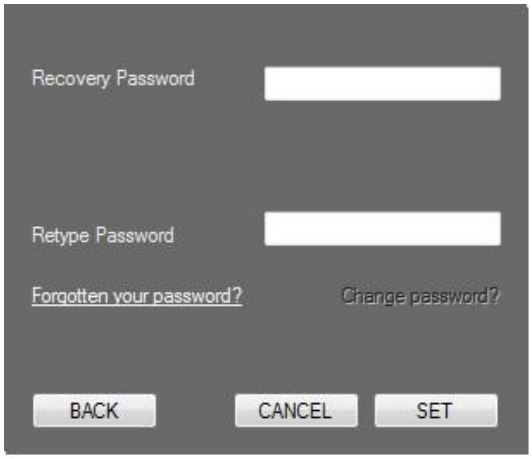
Changing password:

1. Below is the prompt for changing the password. Type the latest password on the first textbox next type your new password then retype it in the next textbox for verification purposes.



Forgotten your password?

1. Initially the application does not have a recovery password so it is a must for the user to set it first. If you click the "Forgotten your password" link the prompt below will appear. After filling up the textboxes click "SET" button.



2. If you already have a recovery password the prompt below will appear if you click the "Forgotten your password?" link. Fill with appropriate recovery password then click "RESET" button.

Troubleshooting Guides and Procedure

1. If the software does not open

    1.1 Unplug the adalpter from the 220 V outlet

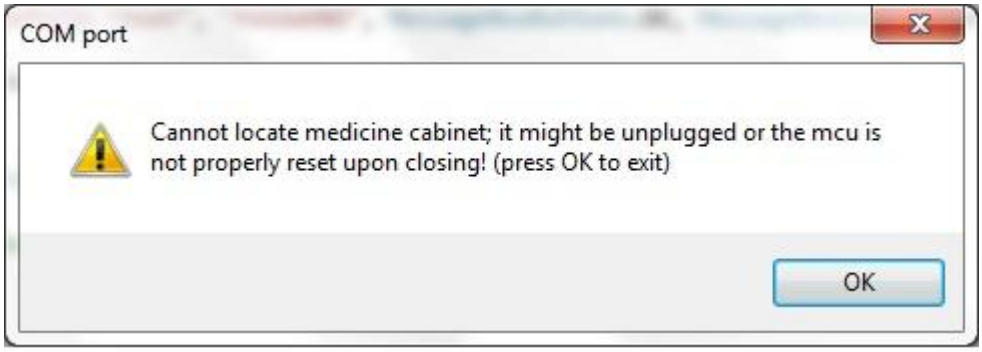
2. If the cabinet does not found by the software

    2.1 Check the device manager for any serial ports available if none seek for technical support.

3. Check if the source outlet is 220 V AC.

Error Definitions

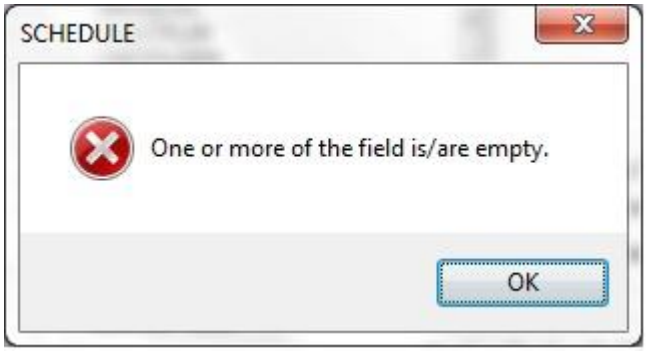
1. Adapter Malfuntioning / Serial port malfunctioning / Device malfunctioning

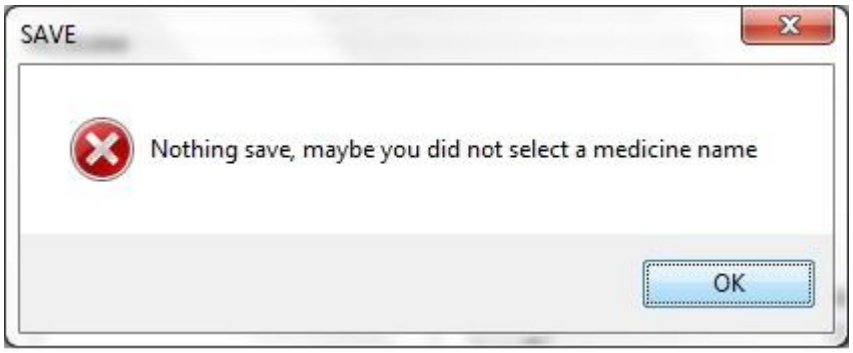
- 
COM port
Cannot locate medicine cabinet; it might be unplugged or the mcu is not properly reset upon closing! (press OK to exit)
OK

2. Wrong data entry / Null data entry

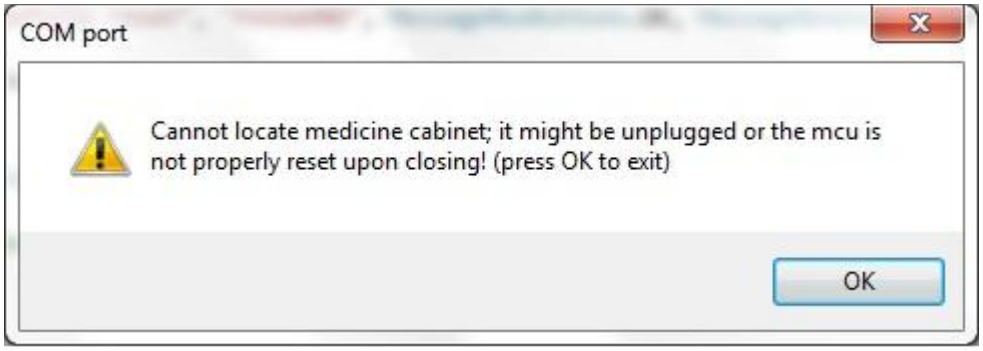
-

-

- 

- 

3. Wrong connections of ports

- 
COM port
Cannot locate medicine cabinet; it might be unplugged or the mcu is not properly reset upon closing! (press OK to exit)
OK

B. Pictures of Prototype

Figure A.3 Complete Set-up

Figure A.4 Complete Set-up with LEDs in on state

Figure A.5 AC Power Adaptor Unregulated.

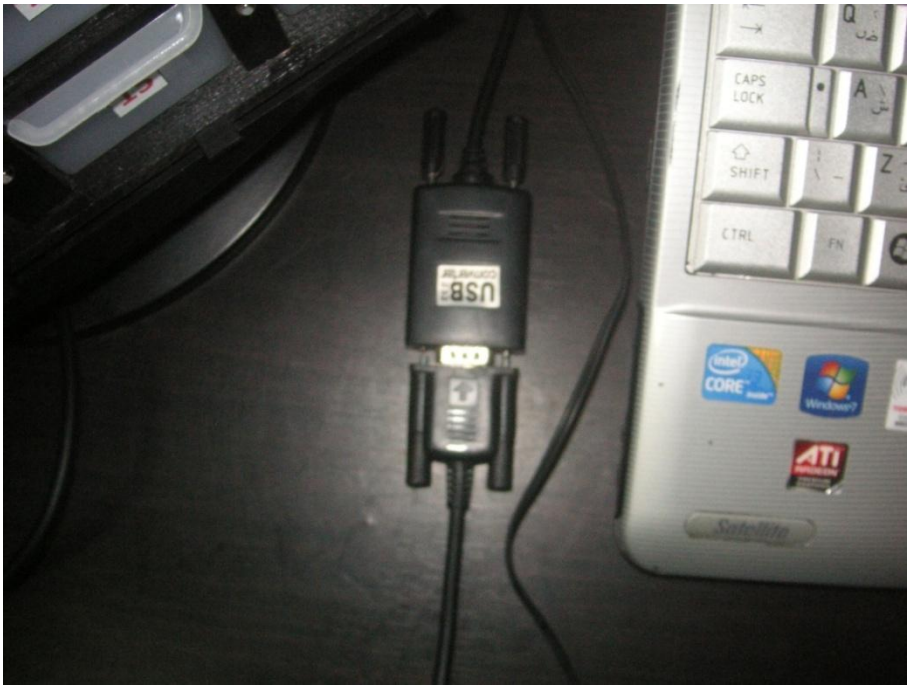


Figure A.6 RS232 Female (below) attach to USB-to-RS232 Male converter (above)

Figure A.7 Medicine Cabinet Front View

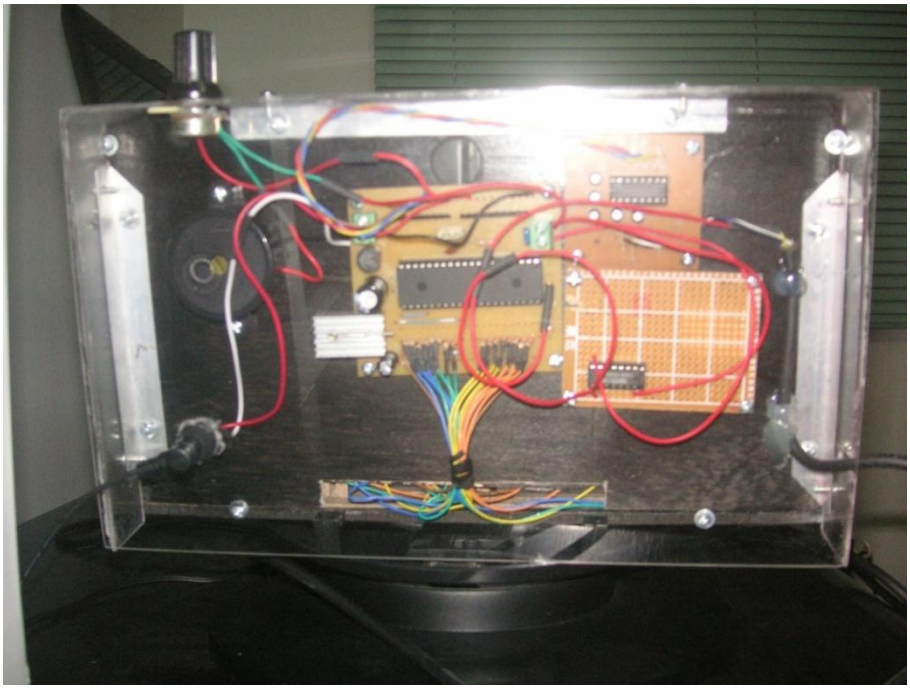


Figure A.8 Rear View of Medicine Cabinet

Figure A.9 Front View of Drawer Number Two

Figure A.10 Top View of Drawer Number Two with Medicine Tablets

Figure A.11 Top View of Drawer Number Two with Medicine in a Wrapper

PIC16F87X
Data Sheet

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

© 2001 Microchip Technology Inc. DS30292C

"All rights reserved. Copyright © 2001, Microchip Technology Incorporated, USA. Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights."

Trademarks

The Microchip name, logo, PIC, PICmicro, PICSTART, PIC MASTER, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, FlexROM, AnalogLAB, MPASM, MPLINK, MPLIB, PICDEM, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR and SelectMode are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2001, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

© 2001 Microchip Technology Inc.

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

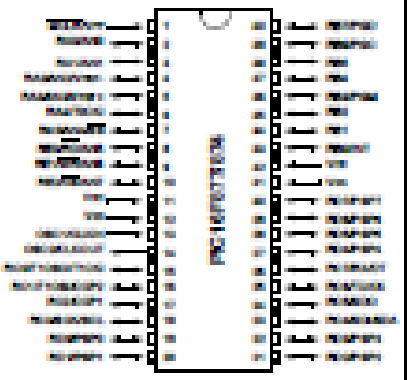
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP

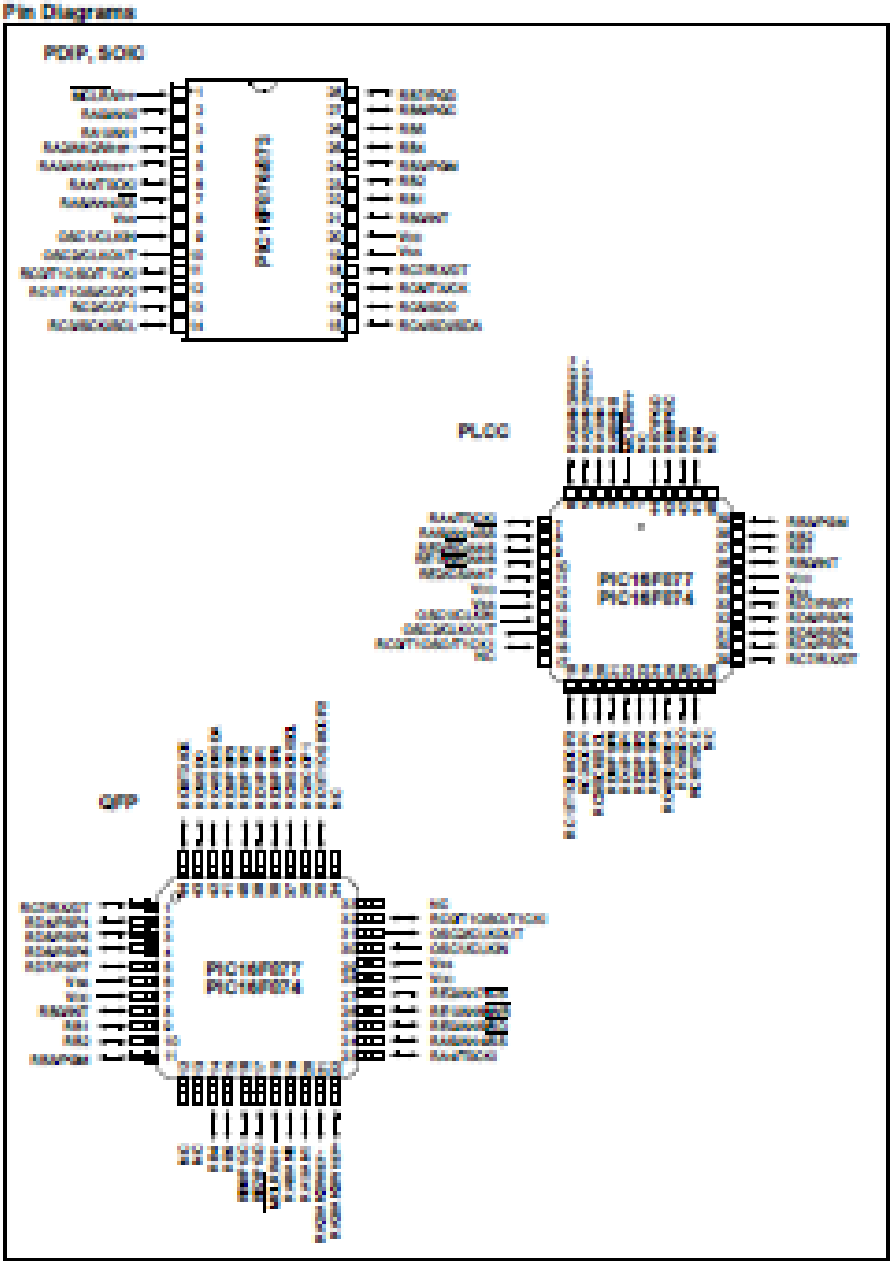


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

© 2001 Microchip Technology Inc.

Pin Diagrams



© 2001 Microchip Technology Inc.

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

© 2001 Microchip Technology Inc.

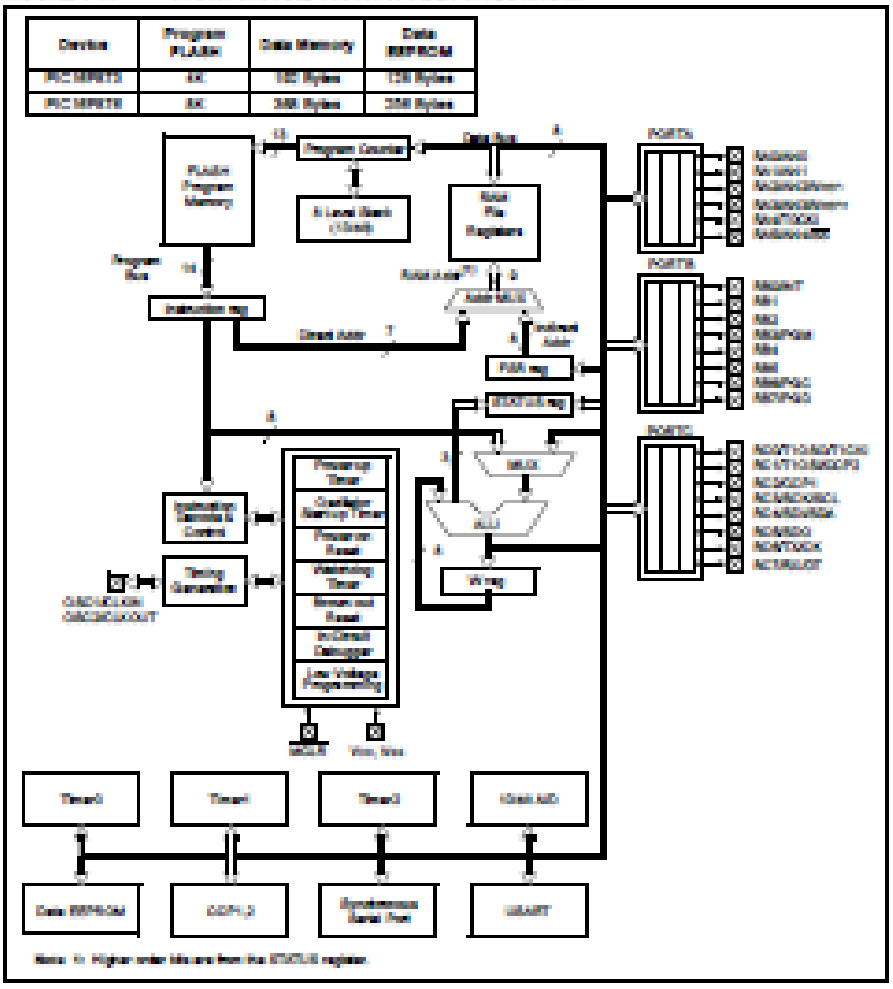
1.0 DEVICE OVERVIEW

This document contains device specific information. Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

There are four devices (PIC16F873, PIC16F874, PIC16F876 and PIC16F877) covered by this data sheet. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.

The following device block diagrams are sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

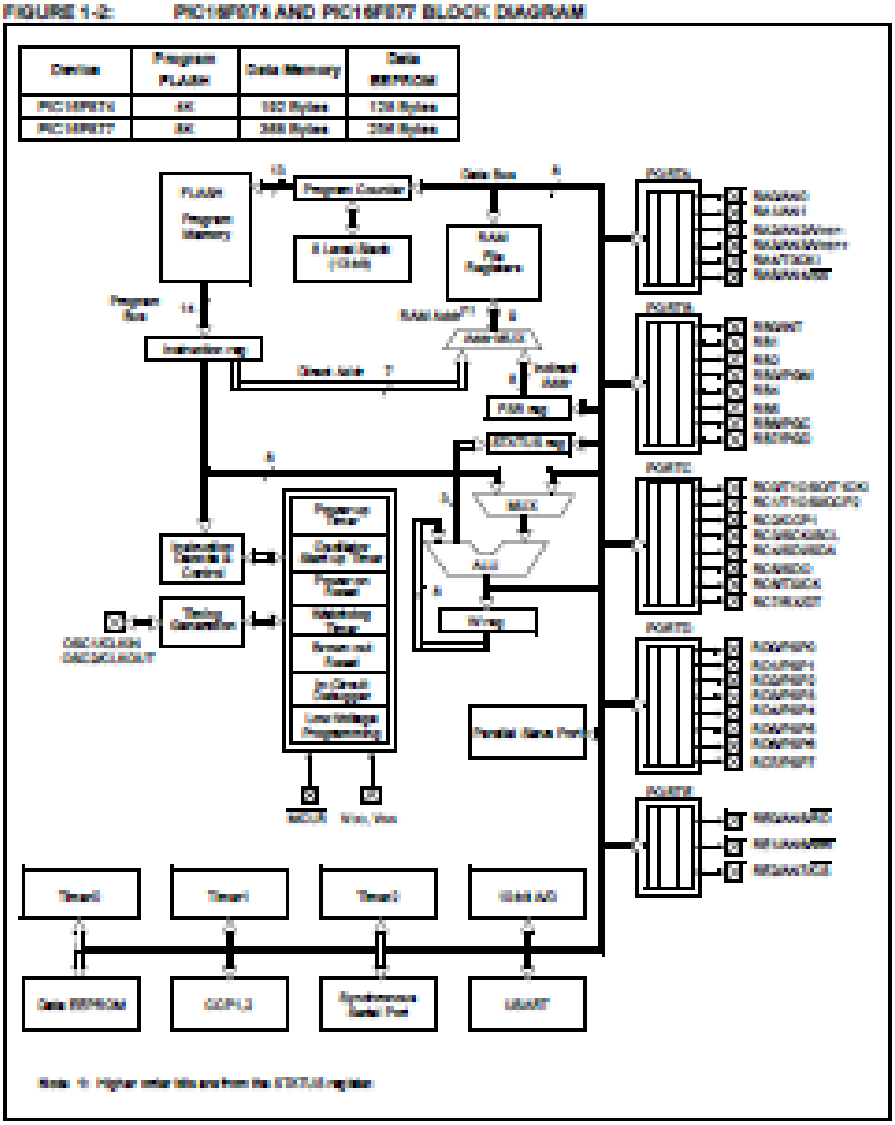
FIGURE 1-1: PIC16F873 AND PIC16F876 BLOCK DIAGRAM



Note: I/O Higher order bits are from the STATUS register.

© 2001 Microchip Technology Inc.

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM



© 2001 Microchip Technology Inc.

TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/SS/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

© 2001 Microchip Technology Inc.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/SS/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

© 2001 Microchip Technology Inc.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL[3]	
RD1/PSP1	20	22	39	I/O	ST/TTL[3]	
RD2/PSP2	21	23	40	I/O	ST/TTL[3]	
RD3/PSP3	22	24	41	I/O	ST/TTL[3]	
RD4/PSP4	27	30	2	I/O	ST/TTL[3]	
RD5/PSP5	28	31	3	I/O	ST/TTL[3]	
RD6/PSP6	29	32	4	I/O	ST/TTL[3]	
RD7/PSP7	30	33	5	I/O	ST/TTL[3]	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL[3]	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL[3]	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL[3]	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

19-4323; Rev 3; 1/99

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers
Low-Power Modems
Interface Translation
Battery-Powered RS-232 Systems
Multi-Drop RS-232 Networks

Features

Superior to Bipolar

- **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX238)**
- **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- **Meet All EIA/TIA-232E and V.28 Specifications**
- **Multiple Drivers and Receivers**
- **3-State Driver and Receiver Outputs**
- **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.
**Contact factory for dice specifications.*

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (µF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	4.7/10	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✓	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✓	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1489
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✓	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✓	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✓	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✓	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✓	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✓	120	Available in quad flatpack package

For free samples & the latest literature: http://www.maxim-ic.com, or phone 1-800-998-8800. For small orders, phone 1-800-835-8769.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
T_{IN} ..... -0.3V to (V_{CC} + 0.3V)
R_{IN} (Except MAX220) ..... ±30V
R_{IN} (MAX220) ..... ±25V
T_{OUT} (Except MAX220) (Note 1) ..... ±15V
T_{OUT} (MAX220) ..... ±13.2V
Output Voltages
T_{OUT} ..... ±15V
R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) 440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C) 842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ..... 0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ..... -40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _ ..... -55°C to +125°C
Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High			2	1.4		V
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate	All except MAX220, normal operation			200	116	kbits/sec
	MAX220			22	20	
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V			±7	±22	mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

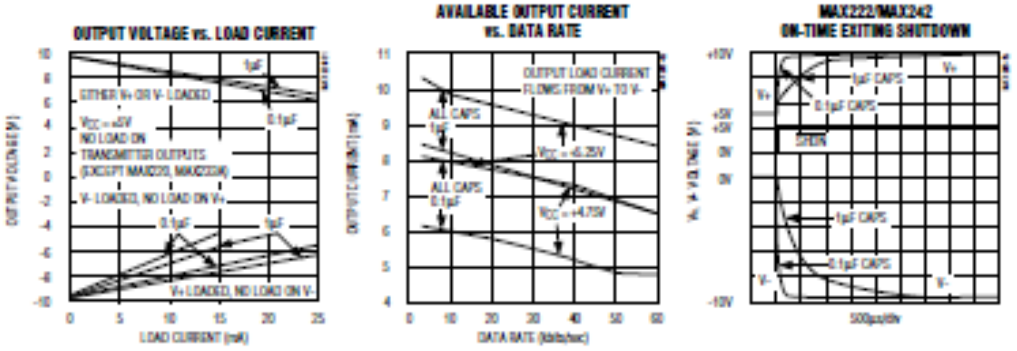
PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

Typical Operating Characteristics

MAX220/MAX222/MAX232A/MAX233A/MAX242/MAX243



National Semiconductor

February 1995

LM78XX Series Voltage Regulators

General Description

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and currents.

The LM78XX series is available in an aluminum TO-3 package which will allow over 1.0A load current if adequate heat sinking is provided. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistor is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating.

Considerable effort was expended to make the LM78XX series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

For output voltage other than 5V, 12V and 15V the LM117 series provides an output voltage range from 1.2V to 57V.

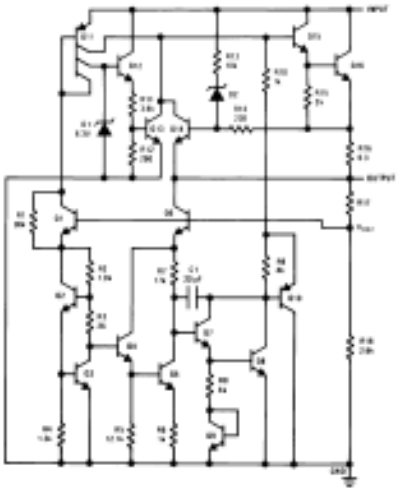
Features

- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

Schematic and Connection Diagrams



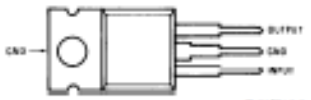
Metal Can Package
TO-3 (K)
Aluminum

TL/H/7744-2

Bottom View

Order Number LM7805CK, LM7812CK or LM7815CK
See NS Package Number KC02A

Plastic Package
TO-220 (T)



TL/H/7744-3

Top View

Order Number LM7805CT, LM7812CT or LM7815CT
See NS Package Number T03B

TL/H/7744-1

©National Semiconductor Corporation TL/H/7744

RRD-B30M105/Printed in U.S.A.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Input Voltage (V_O = 5V, 12V and 15V)	35V
Internal Power Dissipation (Note 1)	Internally Limited
Operating Temperature Range (T_A)	0°C to +70°C
Maximum Junction Temperature (KP package)	150°C
(T Package)	150°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	
TO-3 Package K	300°C
TO-220 Package T	230°C

Electrical Characteristics LM78XXC (Note 2) 0°C ≤ T_J ≤ 125°C unless otherwise noted.

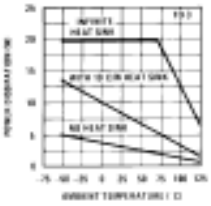
Output Voltage				5V			12V			15V			Units
Input Voltage (unless otherwise noted)				10V			19V			23V			
Symbol	Parameter	Conditions		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_O	Output Voltage	T_J = 25°C, 5 mA ≤ I_O ≤ 1A		4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
		P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A, V_{MIN} ≤ V_{IN} ≤ V_{MAX}		4.75 (7.5 ≤ V_{IN} ≤ 20)		5.25	11.4 (14.5 ≤ V_{IN} ≤ 27)		12.6	14.25 (17.5 ≤ V_{IN} ≤ 30)		15.75	V
ΔV_O	Line Regulation	I_O = 500 mA	T_J = 25°C, ΔV_{IN}		3 (7 ≤ V_{IN} ≤ 25)	50		4 14.5 ≤ V_{IN} ≤ 30)	120		4 (17.5 ≤ V_{IN} ≤ 30)	150	mV
			0°C ≤ T_J ≤ +125°C, ΔV_{IN}			50 (8 ≤ V_{IN} ≤ 20)			120 (15 ≤ V_{IN} ≤ 27)			150 (18.5 ≤ V_{IN} ≤ 30)	mV
		I_O ≤ 1A	T_J = 25°C, ΔV_{IN}			50 (7.5 ≤ V_{IN} ≤ 20)			120 (14.6 ≤ V_{IN} ≤ 27)			150 (17.7 ≤ V_{IN} ≤ 30)	mV
			0°C ≤ T_J ≤ +125°C, ΔV_{IN}			25 (8 ≤ V_{IN} ≤ 12)			60 (16 ≤ V_{IN} ≤ 22)			75 (20 ≤ V_{IN} ≤ 26)	mV
ΔV_O	Load Regulation	T_J = 25°C	5 mA ≤ I_O ≤ 1.5A		10	50		12	120		12	150	mV
			250 mA ≤ I_O ≤ 750 mA			25			60			75	mV
		5 mA ≤ I_O ≤ 1A, 0°C ≤ T_J ≤ +125°C				50			120			150	mV
I_Q	Quiescent Current	I_O ≤ 1A	T_J = 25°C			8			8			8	mA
			0°C ≤ T_J ≤ +125°C			8.5			8.5			8.5	mA
ΔI_Q	Quiescent Current Change	5 mA ≤ I_O ≤ 1A				0.5			0.5			0.5	mA
		T_J = 25°C, I_O ≤ 1A, V_{MIN} ≤ V_{IN} ≤ V_{MAX}				1.0 (7.5 ≤ V_{IN} ≤ 20)			1.0 (14.6 ≤ V_{IN} ≤ 27)			1.0 (17.9 ≤ V_{IN} ≤ 30)	mA V
		I_O ≤ 500 mA, 0°C ≤ T_J ≤ +125°C, V_{MIN} ≤ V_{IN} ≤ V_{MAX}				1.0 (7 ≤ V_{IN} ≤ 25)			1.0 (14.5 ≤ V_{IN} ≤ 30)			1.0 (17.5 ≤ V_{IN} ≤ 30)	mA V
V_N	Output Noise Voltage	T_A = 25°C, 10 Hz ≤ f ≤ 100 kHz			40			75			90		μV
$\frac{\Delta V_{IN}}{\Delta V_{OUT}}$	Ripple Rejection	f = 120 Hz	I_O ≤ 1A, T_J = 25°C or I_O ≤ 500 mA, 0°C ≤ T_J ≤ +125°C	62	80		55	72		54	70		dB
				62			55			54			dB
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}			(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)		V
R_O	Dropout Voltage	T_J = 25°C, I_{OUT} = 1A			2.0			2.0			2.0		V
	Output Resistance	f = 1 kHz			8			18			19		mΩ
	Short-Circuit Current	T_J = 25°C			2.1			1.5			1.2		A
	Peak Output Current	T_J = 25°C			2.4			2.4			2.4		A
	Average TC of V_{OUT}	0°C ≤ T_J ≤ +125°C, I_O = 5 mA			0.6			1.5			1.8		mV/°C
V_{IN}	Input Voltage Required to Maintain Line Regulation	T_J = 25°C, I_O ≤ 1A			7.5		14.6			17.7			V

Note 1: Thermal resistance of the TO-3 package (K, KC) is typically 4°C/W junction to case and 35°C/W case to ambient. Thermal resistance of the TO-220 package (T) is typically 4°C/W junction to case and 50°C/W case to ambient.

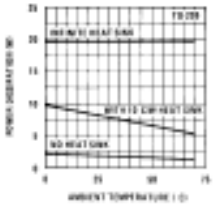
Note 2: All characteristics are measured with capacitor across the input of 0.22 μF, and a capacitor across the output of 0.1 μF. All characteristics except noise voltage and ripple rejection ratio are measured using pulse techniques (t_W ≤ 10 ms, duty cycle ≤ 5%). Output voltage changes due to changes in internal temperature must be taken into account separately.

Typical Performance Characteristics

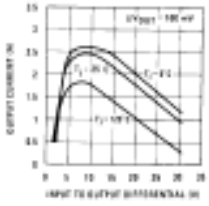
Maximum Average Power Dissipation



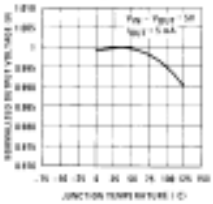
Maximum Average Power Dissipation



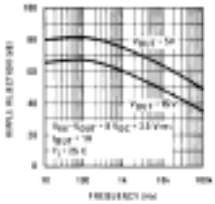
Peak Output Current



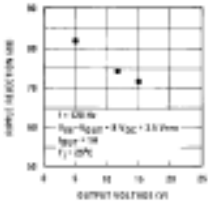
Output Voltage (Normalized to 1V at TJ = 25°C)



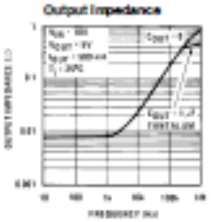
Ripple Rejection



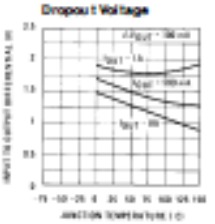
Ripple Rejection



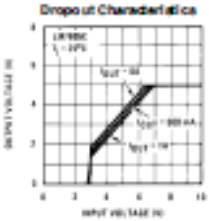
Output Impedance



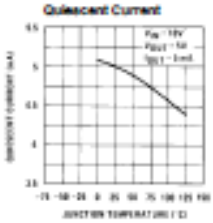
Dropout Voltage



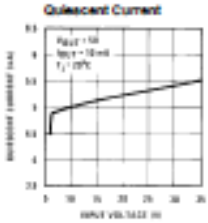
Dropout Characteristics



Quiescent Current



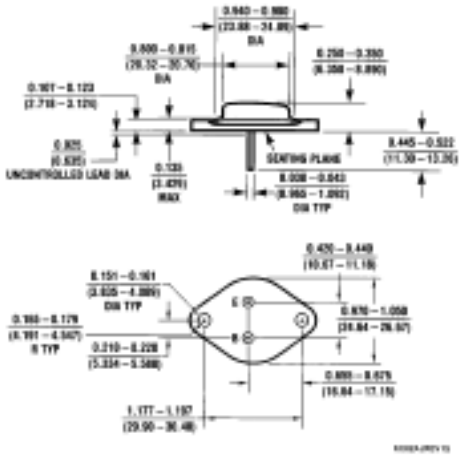
Quiescent Current



TL/H/7754–4

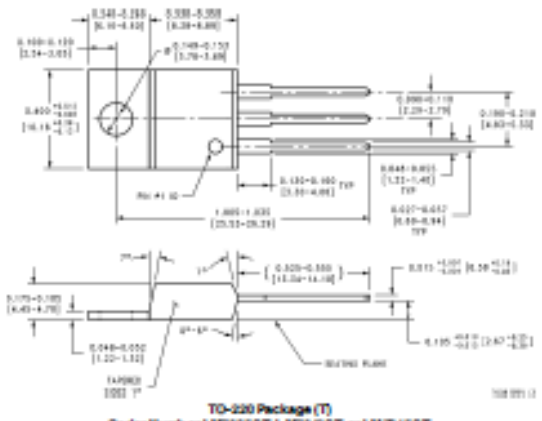
3

Physical Dimensions inches(millimeters)



Aluminum Metal Can Package (KC)
Order Number LM 185SCK, LM 285CK or LM 385SCK
NS Package Number KC02A

Physical Dimensions inches(millimeters) (Continued)



TO-220 Package (T)
Order Number LM7805CT, LM7812CT or LM7815CT
NS Package Number T03B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: onjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

National Semiconductor reserves the right to make changes in the circuitry and specifications contained herein at any time without notice. National Semiconductor makes no representations that the circuitry described herein is free from patent infringement and makes no warranty, express, implied or statutory, including but not limited to the implied warranties of merchantability and fitness for a particular purpose with respect to the use of any integrated circuit described herein. National Semiconductor does not assume any responsibility for use of any circuitry described herein.

D. Program Listing

Mikro C v3.2 Program Listing

```
char read, error;
int i = 0x00, j = 0x00, q = 0;

void main()
{
        ADCON1=6;
        CCP1CON=0x00;

        TRISB = 0x00; // port b as output
        TRISD = 0x00; // port d as output
        TRISE = 0x00; // port e as output

        PORTB = 0x00;          // turn off LED
        PORTD = 0x00;          // turn off LED
        PORTE = 0x00;// turn off PORT E

        UART1_Init(9600);      // Initialize UART module at 9600 bps
        Delay_ms(100);         // Wait for UART module to stabilize

        while (1) // Endless loop
        {
          if (UART1_Data_Ready())      // If data is received
          {
                read = UART1_Read(); // read the received data
                if (read == 's')
                {
                        UART1_Write('x');
                        break;// exit loop
                }
          }
        }

        while(1)
        {
        start:
                if (UART1_Data_Ready())         // If data is received
                {
                        read = UART1_Read();  // read the received data

                        switch(read)
                        {
                                case 'a':   {PORTD = i;
                                        RD0_bit = 1;
                                        i = PORTD;}
                                        break;
                                case 'A':   {PORTD = i;
                                        RD0_bit = 0;
                                        i = PORTD;}
```

```
            break;
case 'b':   {PORTD = i;
            RD1_bit = 1;
            i = PORTD;}
            break;
case 'B':   {PORTD = i;
            RD1_bit = 0;
            i = PORTD;}
            break;
case 'c':   {PORTD = i;
            RD2_bit = 1;
            i = PORTD;}
            break;
case 'C':   {PORTD = i;
            RD2_bit = 0;
            i = PORTD;}
            break;
case 'd':   {PORTD = i;
            RD3_bit = 1;
            i = PORTD;}
            break;
case 'D':   {PORTD = i;
            RD3_bit = 0;
            i = PORTD;}
            break;
case 'e':   {PORTD = i;
            RD4_bit = 1;
            i = PORTD;}
            break;
case 'E':   {PORTD = i;
            RD4_bit = 0;
            i = PORTD;}
            break;
case 'f':   {PORTD = i;
            RD5_bit = 1;
            i = PORTD;}
            break;
case 'F':   {PORTD = i;
            RD5_bit = 0;
            i = PORTD;}
            break;
case 'g':   {PORTD = i;
            RD6_bit = 1;
            i = PORTD;}
            break;
case 'G':   {PORTD = i;
            RD6_bit = 0;
            i = PORTD;}
            break;
case 'h':   {PORTD = i;
            RD7_bit = 1;
            i = PORTD;}
            break;
```

```
case 'H':   {PORTD = i;
             RD7_bit = 0;
             i = PORTD;}
             break;
case 'i':   {PORTB = j;
             RB0_bit = 1;
             j = PORTB;}
             break;
case 'I':   {PORTB = j;
             RB0_bit = 0;
             j = PORTB;}
             break;
case 'j':   {PORTB = j;
             RB1_bit = 1;
             j = PORTB;}
             break;
case 'J':   {PORTB = j;
             RB1_bit = 0;
             j = PORTB;}
             break;
case 'k':   {PORTB = j;
             RB2_bit = 1;
             j = PORTB;}
             break;
case 'K':   {PORTB = j;
             RB2_bit = 0;
             j = PORTB;}
             break;
case 'l':   {PORTB = j;
             RB3_bit = 1;
             j = PORTB;}
             break;
case 'L':   {PORTB = j;
             RB3_bit = 0;
             j = PORTB;}
             break;
case 'm':   {PORTB = j;
             RB4_bit = 1;
             j = PORTB;}
             break;
case 'M':   {PORTB = j;
             RB4_bit = 0;
             j = PORTB;}
             break;
case 'n':   {PORTB = j;
             RB5_bit = 1;
             j = PORTB;}
             break;
case 'N':   {PORTB = j;
             RB5_bit = 0;
             j = PORTB;}
             break;
case 'o':   {PORTB = j;
```

```
                                        RB6_bit = 1;
                                        j = PORTB;}
                                        break;
                                case 'O':   {PORTB = j;
                                        RB6_bit = 0;
                                        j = PORTB;}
                                        break;
                                case 'p':   {PORTB = j;
                                        RB7_bit = 1;
                                        j = PORTB;}
                                        break;
                                case 'P':   {PORTB = j;
                                        RB7_bit = 0;
                                        j = PORTB;}
                                        break;
                                case 'q':   {RE2_bit = 1;}
                                        break;
                                case 'Q':   {RE2_bit = 0;}
                                        break;
                                default:    {goto start;}
                        }
                }
        }
}
```

Microsoft Visual Studio 2010 C# Program Listing

FORM1:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO.Ports;
using System.Threading;
using System.Collections.ObjectModel;
using System.Diagnostics;


namespace WindowsFormsApplication8
{
   public partial class Form1 : Form
   {
      string[] PORTNAME = SerialPort.GetPortNames();
      string drw_no;
      char READ_DATA_FROM_MCU;
      int PORTNAME_INDEX = 0, NOTIFY_ARRAY_INDEX = 0;
      bool search;
      DateTime sched_time;
```

```
                Collection<string> notify_array = new Collection<string>();
                Thread crossThread = null;
                delegate void SetTextCallback();
        private delegate void RefreshTabPageDelegate(int tabPage);

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

this.logTableAdapter.Fill(this._C__PROGRAM_FILES_MICROSOFT_SQL_SERVER_MSSQL10_50_M
SSQLSERVER_MSSQL_DATA_MEDCAB_MDFDataSet1.log);

this.cabstorageTableAdapter.Fill(this._C__PROGRAM_FILES_MICROSOFT_SQL_SERVER_MSSQL1
0_50_MSSQLSERVER_MSSQL_DATA_MEDCAB_MDFDataSet.cabstorage);
            foreach (string s in PORTNAME)
            {
                try
                {
                    port.PortName = s;
                    port.ReadTimeout = 100;
                    port.Open();
                    port.Write("s");
                    READ_DATA_FROM_MCU = (char)port.ReadChar(); //read data from PIC16F877A
then stored to buffer
                    if (READ_DATA_FROM_MCU == 'x')
                    {
                        MessageBox.Show("You may now use the application the medicine cabinet is
connected to: " + PORTNAME[PORTNAME_INDEX], "COM port", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        port.Close();
                        break;
                    }
                }
                catch (Exception a)
                {
                }
                port.Close();
                PORTNAME_INDEX++;
            }

            if (READ_DATA_FROM_MCU != 'x')
            {
                DialogResult result = MessageBox.Show("Cannot locate medicine cabinet; it might be
unplugged or the mcu is not properly reset upon closing! (press OK to exit)", "COM port",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                if (result == DialogResult.OK)
                {
                    goto theEnd;
                }
```

```
        }

        SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
        SqlCommand command;
        SqlDataReader read;

        connection.Open();
        command = new SqlCommand();
        command.Connection = connection;
        command.CommandText = "SELECT * FROM dbo.medicine ORDER BY med_name ASC";
        command.ExecuteNonQuery();
        read = command.ExecuteReader();
        while (read.Read())
        {
            medicine_listBox.Items.Add(read["med_name"].ToString());
        }
        read.Close();
        connection.Close();

        search = true;
        backgroundWorker1.RunWorkerAsync();
        timer1.Start();

        goto Go;

    theEnd:
        Close();

    Go:;
    }

    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (READ_DATA_FROM_MCU == 'x')
        {
            port.DtrEnable = true;
            port.Open();
            port.Close();
            port.DtrEnable = false;
            port.Open();
            port.Close();
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        Current_Date_Time1.Text = DateTime.Now.ToString("f");
        Current_Date_Time2.Text = DateTime.Now.ToString("hh:mm tt");
        Current_Date_Time3.Text = DateTime.Now.ToLongDateString();
    }
```

```
        private void tabControl1_Selected(object sender, EventArgs e)
        {
            if (tabControl1.SelectedTab == tabControl1.TabPages["Schedule_tab"])
            {
                Form2 form2 = new Form2();
                form2.ShowDialog();
                if (MgaVar.okbut == true)
                {
                    MgaVar.okbut = false;
                    this.AcceptButton = schedule_but;
                }
                else
                {
                    tabControl1.SelectTab(Notification_tab);
                }

            }
            if (tabControl1.SelectedTab == tabControl1.TabPages["Notification_tab"])
            {
                this.AcceptButton = notification_off_switch_but;
            }
            if (tabControl1.SelectedTab == tabControl1.TabPages["Edit_tab"])
            {
                Form2 form2 = new Form2();
                form2.ShowDialog();
                if (MgaVar.okbut == true)
                {
                    MgaVar.okbut = false;
                    edit_drawer_listBox.SelectedIndex = -1;
                    this.AcceptButton = edit_save_but;
                }
                else
                {
                    tabControl1.SelectTab(Notification_tab);
                }
            }
            if (tabControl1.SelectedTab == tabControl1.TabPages["Inventory_tab"])
            {
                inventory_drawer_listBox.SelectedIndex = -1;
            }
            if (tabControl1.SelectedTab == tabControl1.TabPages["schedule_list_tab"])
            {

cabstorageTableAdapter.Fill(_C__PROGRAM_FILES_MICROSOFT_SQL_SERVER_MSSQL10_50_MS
SQLSERVER_MSSQL_DATA_MEDCAB_MDFDataSet.cabstorage);
                schedule_list_dataGridView.Refresh();
            }
            if (tabControl1.SelectedTab == tabControl1.TabPages["Log_tab"])
            {
                this.AcceptButton = print_but;

logTableAdapter.Fill(_C__PROGRAM_FILES_MICROSOFT_SQL_SERVER_MSSQL10_50_MSSQLSER
VER_MSSQL_DATA_MEDCAB_MDFDataSet1.log);
```

```
            log_dataGridView.Refresh();
        }
    }

//#############MAINTENANCE TAB ##############################
    private void light_test_selectall_checkBox_CheckedChanged(object sender, EventArgs e)
    {
        if (tabControl1.SelectedTab == tabControl1.TabPages["Maintenance_tab"])
        {
            foreach (CheckBox chkbox in light_test_groupBox.Controls)
            {
                chkbox.Checked = light_test_selectall_checkBox.Checked;
            }
        }

        if (tabControl1.SelectedTab == tabControl1.TabPages["Notification_tab"])
        {
            foreach (CheckBox chkbox in notification_off_switch_gruopBox.Controls)
            {
                chkbox.Checked = notification_selectall_checkBox.Checked;
            }
        }
    }

    private void light_test_on_but_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Warning this action will turn on the selected
LED/s on the cabinet! Do you want to continue?", "Light Test", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            foreach (CheckBox chkbox in light_test_groupBox.Controls)
            {
                if (chkbox.Checked == true)
                {
                    port.Open();
                    switch (chkbox.Text)
                    {
                        case "Drawer1": port.Write("a"); break;
                        case "Drawer2": port.Write("b"); break;
                        case "Drawer3": port.Write("c"); break;
                        case "Drawer4": port.Write("d"); break;
                        case "Drawer5": port.Write("e"); break;
                        case "Drawer6": port.Write("f"); break;
                        case "Drawer7": port.Write("g"); break;
                        case "Drawer8": port.Write("h"); break;
                        case "Drawer9": port.Write("i"); break;
                        case "Drawer10": port.Write("j"); break;
                        case "Drawer11": port.Write("k"); break;
                        case "Drawer12": port.Write("l"); break;
                        case "Drawer13": port.Write("m"); break;
                        case "Drawer14": port.Write("n"); break;
                        case "Drawer15": port.Write("o"); break;
```

```
                    case "Drawer16": port.Write("p"); break;
                }
                port.Close();
            }
        }
    }
}

private void light_test_off_but_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Warning this action will turn off the selected
LED/s on the cabinet! Do you want to continue?", "Warning", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
    if (result == DialogResult.Yes)
    {
        if (tabControl1.SelectedTab == tabControl1.TabPages["Maintenance_tab"])
        {
            foreach (CheckBox chkbox in light_test_groupBox.Controls)
            {
                if (chkbox.Checked == true)
                {
                    port.Open();
                    switch (chkbox.Text)
                    {
                        case "Drawer1": port.Write("A"); break;
                        case "Drawer2": port.Write("B"); break;
                        case "Drawer3": port.Write("C"); break;
                        case "Drawer4": port.Write("D"); break;
                        case "Drawer5": port.Write("E"); break;
                        case "Drawer6": port.Write("F"); break;
                        case "Drawer7": port.Write("G"); break;
                        case "Drawer8": port.Write("H"); break;
                        case "Drawer9": port.Write("I"); break;
                        case "Drawer10": port.Write("J"); break;
                        case "Drawer11": port.Write("K"); break;
                        case "Drawer12": port.Write("L"); break;
                        case "Drawer13": port.Write("M"); break;
                        case "Drawer14": port.Write("N"); break;
                        case "Drawer15": port.Write("O"); break;
                        case "Drawer16": port.Write("P"); break;
                    }
                    port.Close();
                }
            }
        }

        if (tabControl1.SelectedTab == tabControl1.TabPages["Notification_tab"])
        {
            foreach (CheckBox chkbox in notification_off_switch_gruopBox.Controls)
            {
                if (chkbox.Checked == true)
                {
                    port.Open();
```

```
                    switch (chkbox.Text)
                    {
                        case "Drawer1": port.Write("A"); break;
                        case "Drawer2": port.Write("B"); break;
                        case "Drawer3": port.Write("C"); break;
                        case "Drawer4": port.Write("D"); break;
                        case "Drawer5": port.Write("E"); break;
                        case "Drawer6": port.Write("F"); break;
                        case "Drawer7": port.Write("G"); break;
                        case "Drawer8": port.Write("H"); break;
                        case "Drawer9": port.Write("I"); break;
                        case "Drawer10": port.Write("J"); break;
                        case "Drawer11": port.Write("K"); break;
                        case "Drawer12": port.Write("L"); break;
                        case "Drawer13": port.Write("M"); break;
                        case "Drawer14": port.Write("N"); break;
                        case "Drawer15": port.Write("O"); break;
                        case "Drawer16": port.Write("P"); break;
                    }
                    port.Close();
                }
            }
        }
    }

    private void reset_hardware_but_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Warning this action will reset the system! Do
you want to continue?", "Reset Hardware", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            PORTNAME_INDEX = 0;
            port.Open();
            port.Close();
            port.DtrEnable = true;
            port.Open();
            port.Close();
            port.DtrEnable = false;
            port.Open();
            port.Close();

            foreach (string s in PORTNAME)
            {
                try
                {
                    port.PortName = s;
                    port.ReadTimeout = 100;
                    port.Open();
                    port.Write("s");
                    READ_DATA_FROM_MCU = (char)port.ReadChar(); //read data from
PIC16F877A then stored to buffer
```

```
                    if (READ_DATA_FROM_MCU == 'x')
                    {
                        MessageBox.Show("You may now use the application the medicine cabinet is
connected to: " + PORTNAME[PORTNAME_INDEX], "COM port", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        port.Close();
                        break;
                    }
                }
                catch (Exception a)
                {
                }
                port.Close();
                PORTNAME_INDEX++;
            }

            if (READ_DATA_FROM_MCU != 'x')
            {
                result = MessageBox.Show("Cannot locate medicine cabinet; it might be unplugged
or the mcu is not properly reset upon closing! (press OK to exit)", "COM port",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                if (result == DialogResult.OK)
                {
                    Close();
                }
            }
        }
    }

    private void show_com_port_number_but_Click(object sender, EventArgs e)
    {
        MessageBox.Show("The medicine cabinet is connected to: " +
PORTNAME[PORTNAME_INDEX], "COM port", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }

    private void buzzer_test_on_but_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Warning this action will turn on the buzzer on
the cabinet! Do you want to continue?", "Buzzer Test", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            port.Open();
            port.Write("q");
            port.Close();
        }
    }

    private void buzzer_test_off_but_Click(object sender, EventArgs e)
    {
```

```
        DialogResult result = MessageBox.Show("Warning this action will turn off the buzzer on
the cabinet! Do you want to continue?", "Buzzer Test", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            port.Open();
            port.Write("Q");
            port.Close();
        }
    }
//######## MAINTENANCE TAB END ###################

//############ SCHEDULE TAB ###################
    private void schedule_but_Click(object sender, EventArgs e)
    {
        int time_interval_in_hours, a_day_in_hours = 24;
        DialogResult result = MessageBox.Show("Are you sure with this action?", "SCHEDULE",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);

        if (result == DialogResult.Yes)
        {
            foreach (Control item in Schedule_tab.Controls)
            {
                if (item.GetType() == typeof(GroupBox))
                {
                    foreach (Control item1 in item.Controls)
                    {
                        if (item1.GetType() == typeof(MaskedTextBox))
                        {
                            if (item1.Name == "medicine_name")
                            {
                                continue;
                            }

                            else if (string.IsNullOrEmpty(((MaskedTextBox)item1).Text))
                            {
                                result = MessageBox.Show("One or more of the field is/are empty.",
"SCHEDULE", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                goto proceed;
                            }
                        }

                        else if (item1.GetType() == typeof(ComboBox))
                        {
                            if (string.IsNullOrEmpty(((ComboBox)item1).Text))
                            {
                                result = MessageBox.Show("One or more of the field is/are empty.",
"SCHEDULE", MessageBoxButtons.OK, MessageBoxIcon.Error);
                                goto proceed;
                            }
                        }
                    }
                }
```

```
            }

        proceed:
            if (medicine_listBox.SelectedItem == null)
            {
                result = MessageBox.Show("Please select a medicine.", "SCHEDULE",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;
            SqlDataReader read;

            command = new SqlCommand();
            command.Connection = connection;
            connection.Open();
            try
            {
                command.CommandText = string.Format("SELECT * FROM dbo.cabstorage WHERE
drawer_no={0}", drawer_number_comboBox.Text);
                command.ExecuteNonQuery();
                read = command.ExecuteReader();
                while (read.Read())
                {
                    if (read["patient_name"].ToString().ToUpper() != string.Format("{0}, {1} {2}",
lastname_patient.Text.Trim(), firstname_patient.Text.Trim(),
middle_patient.Text.Trim()).ToUpper())
                    {
                        MessageBox.Show("The drawer number you have selected is occupied by
another patient. Please select another drawer number", "SCHEDULE", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        read.Close();
                        goto proceed1;
                    }

                    if (read["med_name"].ToString() == medicine_listBox.SelectedItem.ToString())
                    {
                        MessageBox.Show("The medicine name was already assigned to this patient.
Please select another medicine", "SCHEDULE", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        read.Close();
                        goto proceed1;
                    }
                }
                read.Close();
                connection.Close();
            }
            catch
            {
                goto proceed1;
            }
```

```
            if (result != DialogResult.OK)
            {
                time_interval_in_hours = a_day_in_hours /
int.Parse(no_of_intake_per_day_comboBox.Text);
                sched_time = DateTime.Parse(starting_time_medicine.Value.ToString("HH:mm"));

                while (a_day_in_hours != 0)
                {
                    command = new SqlCommand();
                    command.Connection = connection;
                    connection.Open();
                    command.CommandText = "INSERT INTO cabstorage(drawer_no,
patient_name, assigned_nurse, med_name, qty_per_intake, time_intake, date_of_schedule,
room_no, every_other_day, total_qty, date_of_intake) VALUES(@_drawer_no, @_patient_name,
@_assigned_nurse, @_med_name, @_qty_per_intake, @_time_intake, @_date_of_schedule,
@_room_no, @_every_other_day, @_total_qty, @_date_of_intake)";

                    command.Parameters.AddWithValue("@_drawer_no",
int.Parse(drawer_number_comboBox.Text));
                    command.Parameters.AddWithValue("@_patient_name", string.Format("{0},
{1} {2}", lastname_patient.Text.Trim(), firstname_patient.Text.Trim(),
middle_patient.Text.Trim()));
                    command.Parameters.AddWithValue("@_assigned_nurse", string.Format("{0},
{1} {2}", lastname_nurse.Text.Trim(), firstname_nurse.Text.Trim(), middle_nurse.Text.Trim()));
                    command.Parameters.AddWithValue("@_room_no", int.Parse(room_no.Text));
                    command.Parameters.AddWithValue("@_med_name",
medicine_listBox.SelectedItem.ToString());
                    command.Parameters.AddWithValue("@_total_qty",
int.Parse(medicine_total_qty.Text));
                    command.Parameters.AddWithValue("@_qty_per_intake",
int.Parse(medicine_qty_per_intake.Text));
                    command.Parameters.AddWithValue("@_time_intake", sched_time);
                    command.Parameters.AddWithValue("@_date_of_schedule",
DateTime.Parse(Convert.ToString(DateTime.Now.Date).Substring(0, 9)));
                    command.Parameters.AddWithValue("@_date_of_intake",
DateTime.Parse(Convert.ToString(DateTime.Now.Date).Substring(0, 9)));
                    command.Parameters.AddWithValue("@_every_other_day",
(medicine_every_day_radioButton.Checked == false) ? "T" : "");
                    command.ExecuteNonQuery();
                    connection.Close();

                    sched_time = sched_time.AddHours(time_interval_in_hours);
                    a_day_in_hours -= time_interval_in_hours;
                }
                a_day_in_hours = 24;
            }
        proceed1:
            connection.Close();
        }
    }

    private void add_medicine_but_Click(object sender, EventArgs e)
    {
```

```
        DialogResult result = MessageBox.Show("Are you sure you want to do this?", "ADD
MEDICINE", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
        {
            if (!string.IsNullOrEmpty(medicine_name.Text))
            {
                SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
                SqlCommand command;
                SqlDataReader read;

                try
                {
                    connection.Open();
                    command = new SqlCommand();
                    command.Connection = connection;
                    command.CommandText = "INSERT INTO dbo.medicine(med_name)
VALUES(@_med_name)";
                    command.Parameters.AddWithValue("@_med_name",
medicine_name.Text.ToUpper().Trim());
                    command.ExecuteNonQuery();
                    command.CommandText = "SELECT * FROM dbo.medicine ORDER BY
med_name ASC";
                    command.ExecuteNonQuery();
                    read = command.ExecuteReader();
                    medicine_listBox.Items.Clear();
                    while (read.Read())
                    {
                        medicine_listBox.Items.Add(read["med_name"].ToString());
                    }
                    read.Close();
                    connection.Close();
                }
                catch
                {
                    MessageBox.Show("Medicine name already exists", "ADD MEDICINE",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                }
            }
            else
            {
                MessageBox.Show("Please enter a medicine name.", "ADD MEDICINE",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }

    private void delete_medicine_but_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("Are you sure you want to do this?", "DELETE
MEDICINE", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
```

```
        {
            if (medicine_listBox.SelectedItem != null)
            {
                SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
                SqlCommand command;

                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = string.Format("DELETE FROM dbo.medicine WHERE
med_name='{0}'", medicine_listBox.SelectedItem.ToString());
                command.ExecuteNonQuery();
                connection.Close();

                medicine_listBox.Items.Remove(medicine_listBox.SelectedItem);
            }
            else
            {
                MessageBox.Show("Please selesct a medicine name.", "DELETE MEDICINE",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }

    private void lastname_patient_Click(object sender, EventArgs e)
    {
        if (sender.Equals(room_no))
        {
            this.room_no.Select(0, 0);
        }
        else if (sender.Equals(medicine_total_qty))
        {
            this.medicine_total_qty.Select(0, 0);
        }
        else if (sender.Equals(medicine_qty_per_intake))
        {
            this.medicine_qty_per_intake.Select(0, 0);
        }
        else if (sender.Equals(edit_total_quantity))
        {
            this.edit_total_quantity.Select(0, 0);
        }
        else if (sender.Equals(edit_quantity_per_intake))
        {
            this.edit_quantity_per_intake.Select(0, 0);
        }
    }

    private void clear_but_Click(object sender, EventArgs e)
    {
        foreach (Control item in Schedule_tab.Controls)
```

```
            {
                if (item.GetType() == typeof(GroupBox))
                {
                    foreach (Control item1 in item.Controls)
                    {
                        if (item1.GetType() == typeof(MaskedTextBox))
                        {
                            item1.ResetText();
                        }

                        else if (item1.GetType() == typeof(ComboBox))
                        {
                            ComboBox dd = (ComboBox)item1;
                            dd.SelectedIndex = -1;
                        }
                    }
                }
            }
        }
//######### SCHEDULE TAB END ############################

//########## NOTIFICATION TAB ##################
        private void notification_clear_but_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to erase all the
notifications?", "CLEAR NOTIFICATION/S", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {
                SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
                SqlCommand command;

                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = "DELETE FROM dbo.notify";
                command.ExecuteNonQuery();
                connection.Close();

                notification_listbox.Items.Clear();
            }
        }
//############ NOTIFICATION TAB END ################################

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            while (search)
            {
                Collection<string> retrieve_date_of_intake = new Collection<string>(),
                                   retrieve_access_id = new Collection<string>(),
                                   retrieve_every_other_day = new Collection<string>(),
```

```
                        retrieve_drawer_no = new Collection<string>(),
                        retrieve_patient_name = new Collection<string>(),
                        retrieve_assigned_nurse = new Collection<string>(),
                        retrieve_room_no = new Collection<string>(),
                        retrieve_med_name = new Collection<string>(),
                        retrieve_qty_per_intake = new Collection<string>(),
                        retrieve_time_intake = new Collection<string>(),
                        retrieve_date_of_schedule = new Collection<string>(),
                        retrieve_total_qty = new Collection<string>();


            int c1 = 0, c2 = 0, i = 0;

            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;
            SqlDataReader read;

            connection.Open();
            command = new SqlCommand();
            command.Connection = connection;
            command.CommandText = string.Format("SELECT * FROM dbo.cabstorage WHERE
time_intake='{0}' AND date_of_intake='{1}'", DateTime.Now.ToString("HH:mm"),
DateTime.Now.ToString("yyyy-MMM-dd"));
            command.ExecuteNonQuery();
            read = command.ExecuteReader();
            while (read.Read())
            {
                retrieve_access_id.Add(read["access_id"].ToString());
                retrieve_every_other_day.Add(read["every_other_day"].ToString());

                retrieve_drawer_no.Add(read["drawer_no"].ToString());

retrieve_date_of_intake.Add(DateTime.Parse(read["date_of_intake"].ToString()).ToString("yyyy-
MMM-dd"));
                retrieve_patient_name.Add(read["patient_name"].ToString());
                retrieve_assigned_nurse.Add(read["assigned_nurse"].ToString());
                retrieve_room_no.Add(read["room_no"].ToString());
                retrieve_med_name.Add(read["med_name"].ToString());
                retrieve_qty_per_intake.Add(read["qty_per_intake"].ToString());

retrieve_time_intake.Add(DateTime.Parse(read["time_intake"].ToString()).ToString("HH:mm"));

retrieve_date_of_schedule.Add(DateTime.Parse(read["date_of_schedule"].ToString()).ToString("y
yyy-MMM-dd"));
                retrieve_total_qty.Add(read["total_qty"].ToString());
                c1 = ++c2;
            }
            read.Close();
            connection.Close();

            while (c1 > 0)
```

```
            {
                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = "INSERT INTO dbo.notify(drawer_no, patient_name,
assigned_nurse, med_name, qty_per_intake, time_intake, room_no, date_of_intake)
VALUES(@_drawer_no1, @_patient_name1, @_assigned_nurse1, @_med_name1,
@_qty_per_intake1, @_time_intake1, @_room_no1, @_date_of_intake1)";

                command.Parameters.AddWithValue("@_drawer_no1",
int.Parse(retrieve_drawer_no[i]));
                command.Parameters.AddWithValue("@_patient_name1",
retrieve_patient_name[i]);
                command.Parameters.AddWithValue("@_assigned_nurse1",
retrieve_assigned_nurse[i]);
                command.Parameters.AddWithValue("@_room_no1",
int.Parse(retrieve_room_no[i]));
                command.Parameters.AddWithValue("@_med_name1", retrieve_med_name[i]);
                command.Parameters.AddWithValue("@_qty_per_intake1",
int.Parse(retrieve_qty_per_intake[i]));
                command.Parameters.AddWithValue("@_time_intake1",
DateTime.Parse(retrieve_time_intake[i]).ToString("HH:mm"));
                command.Parameters.AddWithValue("@_date_of_intake1",
DateTime.Parse(retrieve_date_of_intake[i]).ToString("yyyy-MMM-dd"));
                command.ExecuteNonQuery();

                command.CommandText = "INSERT INTO dbo.log(drawer_no, patient_name,
assigned_nurse, med_name, qty_per_intake, time_intake, room_no, date_of_intake,
date_of_schedule) VALUES(@_drawer_no11, @_patient_name11, @_assigned_nurse11,
@_med_name11, @_qty_per_intake11, @_time_intake11, @_room_no11, @_date_of_intake11,
@_date_of_schedule11)";
                command.Parameters.AddWithValue("@_drawer_no11",
int.Parse(retrieve_drawer_no[i]));
                command.Parameters.AddWithValue("@_patient_name11",
retrieve_patient_name[i]);
                command.Parameters.AddWithValue("@_assigned_nurse11",
retrieve_assigned_nurse[i]);
                command.Parameters.AddWithValue("@_room_no11",
int.Parse(retrieve_room_no[i]));
                command.Parameters.AddWithValue("@_med_name11", retrieve_med_name[i]);
                command.Parameters.AddWithValue("@_qty_per_intake11",
int.Parse(retrieve_qty_per_intake[i]));
                command.Parameters.AddWithValue("@_time_intake11",
DateTime.Parse(retrieve_time_intake[i]).ToString("HH:mm"));
                command.Parameters.AddWithValue("@_date_of_intake11",
DateTime.Parse(retrieve_date_of_intake[i]).ToString("yyyy-MMM-dd"));
                command.Parameters.AddWithValue("@_date_of_schedule11",
DateTime.Parse(retrieve_date_of_schedule[i]).ToString("yyyy-MMM-dd"));
                command.ExecuteNonQuery();
                connection.Close();
                i++;
                c1--;
            }
```

```
            if (i > 0)
            {
                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;
                notify_array.Clear();
                this.crossThread = new Thread(new ThreadStart(this.ClearNotification));
                this.crossThread.Start();

                command.CommandText = "SELECT * FROM dbo.notify";
                command.ExecuteNonQuery();
                read = command.ExecuteReader();
                while (read.Read())
                {
                    notify_array.Add("DRAWER" + (read["drawer_no"].ToString()) + ": " +
(read["patient_name"].ToString()) + " MEDICATION: " + (read["med_name"].ToString()) + " " +
(read["qty_per_intake"].ToString()) + " pc/pcs ASSIGNED TO: " +
(read["assigned_nurse"].ToString()) + " TIME: " + (read["time_intake"].ToString().Substring(0,
5)) + " DATE: " + (read["date_of_intake"].ToString().Substring(0, 9)));
                }
                read.Close();

                command.CommandText = "SELECT COUNT(notify_id) FROM dbo.notify";
                NOTIFY_ARRAY_INDEX = (Int32)command.ExecuteScalar();

                this.crossThread = new Thread(new ThreadStart(this.ShowNotificationTab));
                this.crossThread.Start();
                i = 0;
            }

            while (c2 > 0)
            {
                port.Open();
                switch (retrieve_drawer_no[i])
                {
                    case "1": port.Write("a"); break;
                    case "2": port.Write("b"); break;
                    case "3": port.Write("c"); break;
                    case "4": port.Write("d"); break;
                    case "5": port.Write("e"); break;
                    case "6": port.Write("f"); break;
                    case "7": port.Write("g"); break;
                    case "8": port.Write("h"); break;
                    case "9": port.Write("i"); break;
                    case "10": port.Write("j"); break;
                    case "11": port.Write("k"); break;
                    case "12": port.Write("l"); break;
                    case "13": port.Write("m"); break;
                    case "14": port.Write("n"); break;
                    case "15": port.Write("o"); break;
                    case "16": port.Write("p"); break;
                }
```

```
                port.Write("q");
                Thread.Sleep(200);
                port.Write("Q");
                port.Close();

                int var = int.Parse(retrieve_total_qty[i]);
                var -= int.Parse(retrieve_qty_per_intake[i]);
                retrieve_total_qty[i] = var.ToString();

                if (var >= 0)
                {
                    command.CommandText = string.Format("UPDATE dbo.cabstorage SET
total_qty='{0}' WHERE med_name='{1}'", retrieve_total_qty[i], retrieve_med_name[i]);
                    command.ExecuteNonQuery();
                }
                else
                {

                }

                if (retrieve_every_other_day[i] == "")
                {
                    command.CommandText = string.Format("UPDATE dbo.cabstorage SET
date_of_intake='{0}' WHERE access_id='{1}'",
DateTime.Parse(retrieve_date_of_intake[i]).AddDays(1).ToString("yyyy-MMM-dd"),
retrieve_access_id[i]);
                    command.ExecuteNonQuery();
                }
                else if (retrieve_every_other_day[i] == "T")
                {
                    command.CommandText = string.Format("UPDATE dbo.cabstorage SET
date_of_intake='{0}' WHERE access_id='{1}'",
DateTime.Parse(retrieve_date_of_intake[i]).AddDays(2).ToString("yyyy-MMM-dd"),
retrieve_access_id[i]);
                    command.ExecuteNonQuery();
                }
                i++;
                c2--;
            }
            connection.Close();
        }
    }


    private void ShowNotificationTab()
    {
        if (this.InvokeRequired)
        {
            SetTextCallback d = new SetTextCallback(ShowNotificationTab);
            this.Invoke(d, new object[] { });
        }
        else
        {
```

```
            tabControl1.SelectTab(Notification_tab);

            while (NOTIFY_ARRAY_INDEX > 0)
            {
                notification_listbox.Items.Add(notify_array[--NOTIFY_ARRAY_INDEX]);
            }
        }
    }

    private void ClearNotification()
    {
        if (this.InvokeRequired)
        {
            SetTextCallback d = new SetTextCallback(ClearNotification);
            this.Invoke(d, new object[] { });
        }
        else
        {
            notification_listbox.Items.Clear();
        }
    }

//######## EDIT TAB ######################################
    private void edit_drawer_listBox_SelectedValueChanged(object sender, EventArgs e)
    {
        string previous = "\0";
        edit_medicine_listBox.Items.Clear();

        foreach (Control item in Edit_tab.Controls)
        {
            if (item.GetType() == typeof(GroupBox))
            {
                foreach (Control item1 in item.Controls)
                {
                    if (item1.GetType() == typeof(MaskedTextBox))
                    {
                        item1.ResetText();
                    }

                    else if (item1.GetType() == typeof(ComboBox))
                    {
                        ComboBox dd = (ComboBox)item1;
                        dd.SelectedIndex = -1;
                    }

                    else if (item1.GetType() == typeof(RadioButton))
                    {
                        RadioButton dd = (RadioButton)item1;
                        dd.Checked = false;
                    }
                }
            }
        }
```

```
        if (edit_drawer_listBox.SelectedIndex >= 0)
        {
            if (edit_drawer_listBox.SelectedItem.ToString().Length == 8)
                drw_no = edit_drawer_listBox.SelectedItem.ToString().Substring(7, 1);
            else
                drw_no = edit_drawer_listBox.SelectedItem.ToString().Substring(7, 2);

            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;
            SqlDataReader read;

            connection.Open();
            command = new SqlCommand();
            command.Connection = connection;
            command.CommandText = string.Format("SELECT * FROM dbo.cabstorage WHERE
drawer_no='{0}' ORDER BY med_name ASC", drw_no);
            command.ExecuteNonQuery();
            read = command.ExecuteReader();
            while (read.Read())
            {
                edit_patient_full_name.Text = read["patient_name"].ToString();
                edit_room_number.Text = read["room_no"].ToString();
                edit_nurse_full_name.Text = read["assigned_nurse"].ToString();
                if (read["med_name"].ToString() != previous)
                {
                    edit_medicine_listBox.Items.Add(read["med_name"].ToString());
                }
                previous = read["med_name"].ToString();
            }
            read.Close();
            connection.Close();
        }
    }

    private void edit_medicine_listBox_SelectedValueChanged(object sender, EventArgs e)
    {
        SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
        SqlCommand command;
        SqlDataReader read;

        if (edit_medicine_listBox.SelectedItem != null)
        {
            connection.Open();
            command = new SqlCommand();
            command.Connection = connection;
            command.CommandText = string.Format("SELECT COUNT(access_id) FROM
dbo.cabstorage WHERE drawer_no='{0}' AND med_name='{1}'", drw_no,
edit_medicine_listBox.SelectedItem.ToString());
```

```
            int numberofintake = (Int32)command.ExecuteScalar();
            edit_number_per_intake.SelectedIndex = numberofintake - 1;

            command.CommandText = string.Format("SELECT * FROM dbo.cabstorage WHERE
drawer_no='{0}' AND med_name='{1}'", drw_no,
edit_medicine_listBox.SelectedItem.ToString());
            command.ExecuteNonQuery();
            read = command.ExecuteReader();
            read.Read();
            edit_starting_time.Value = DateTime.Parse(read["time_intake"].ToString());
            edit_total_quantity.Text = read["total_qty"].ToString();
            edit_quantity_per_intake.Text = read["qty_per_intake"].ToString();
            if (read["every_other_day"].ToString() == "")
               edit_every_day.Checked = true;
            else
               edit_every_other_day.Checked = true;
            read.Close();
            connection.Close();
         }
      }

      private void edit_save_but_Click(object sender, EventArgs e)
      {
         DialogResult result = MessageBox.Show("This action will affect a schedule of a medicine
for the patient assigned for this drawer. Do you want to continue?", "SAVE",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

         if (result == DialogResult.Yes)
         {
            int kk = 0, interval, edited_number_of_intake;
            Collection<string> access_id_retrieve = new Collection<string>(),
date_intake_retrieve = new Collection<string>(), date_scheduled_retrieve = new
Collection<string>();
            string edited_starting_time, previous_starting_time;

            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;
            SqlDataReader read;

            try
            {
               connection.Open();
               command = new SqlCommand();
               command.Connection = connection;
               command.CommandText = string.Format("UPDATE dbo.cabstorage SET
patient_name='{0}', room_no='{1}' WHERE drawer_no='{2}'", edit_patient_full_name.Text,
edit_room_number.Text, drw_no);
               command.ExecuteNonQuery();

               command.CommandText = string.Format("UPDATE dbo.cabstorage SET
assigned_nurse='{0}', total_qty='{1}', qty_per_intake='{2}', every_other_day='{3}' WHERE
```

```
drawer_no='{4}' AND med_name='{5}'", edit_nurse_full_name.Text, edit_total_quantity.Text,
edit_quantity_per_intake.Text, (edit_every_day.Checked == false) ? "T" : "", drw_no,
edit_medicine_listBox.SelectedItem.ToString());
                command.ExecuteNonQuery();

                command.CommandText = string.Format("SELECT * FROM dbo.cabstorage WHERE
drawer_no='{0}' AND med_name='{1}'", drw_no,
edit_medicine_listBox.SelectedItem.ToString());
                command.ExecuteNonQuery();
                read = command.ExecuteReader();
                while (read.Read())
                {
                   access_id_retrieve.Add(read["access_id"].ToString());
                }
                read.Close();
                connection.Close();

                edited_number_of_intake = int.Parse(edit_number_per_intake.Text);
                edited_starting_time = edit_starting_time.Value.ToString("HH:mm");
                interval = 24 / int.Parse(edit_number_per_intake.Text);
                previous_starting_time = edited_starting_time;

                while (edited_number_of_intake > 0)
                {
                   try
                   {
                      connection.Open();
                      command = new SqlCommand();
                      command.Connection = connection;
                      command.CommandText = string.Format("UPDATE dbo.cabstorage SET
time_intake='{0}' WHERE access_id='{1}'", edited_starting_time, access_id_retrieve[kk]);
                      command.ExecuteNonQuery();
                      connection.Close();
                   }
                   catch
                   {
                      connection.Close();
                      connection.Open();
                      command = new SqlCommand();
                      command.Connection = connection;
                      command.CommandText = "INSERT INTO cabstorage(drawer_no,
patient_name, assigned_nurse, med_name, qty_per_intake, time_intake, date_of_schedule,
room_no, every_other_day, total_qty, date_of_intake) VALUES(@_drawer_no, @_patient_name,
@_assigned_nurse, @_med_name, @_qty_per_intake, @_time_intake, @_date_of_schedule,
@_room_no, @_every_other_day, @_total_qty, @_date_of_intake)";

                      command.Parameters.AddWithValue("@_drawer_no", int.Parse(drw_no));
                      command.Parameters.AddWithValue("@_patient_name",
edit_patient_full_name.Text);
                      command.Parameters.AddWithValue("@_assigned_nurse",
edit_nurse_full_name.Text);
                      command.Parameters.AddWithValue("@_room_no",
int.Parse(edit_room_number.Text));
```

```
                    command.Parameters.AddWithValue("@_med_name",
edit_medicine_listBox.SelectedItem.ToString());
                    command.Parameters.AddWithValue("@_total_qty",
int.Parse(edit_total_quantity.Text));
                    command.Parameters.AddWithValue("@_qty_per_intake",
int.Parse(edit_quantity_per_intake.Text));
                    command.Parameters.AddWithValue("@_time_intake", edited_starting_time);
                    command.Parameters.AddWithValue("@_date_of_schedule",
DateTime.Parse(Convert.ToString(DateTime.Now.Date).Substring(0, 9)));
                    command.Parameters.AddWithValue("@_date_of_intake",
DateTime.Parse(Convert.ToString(DateTime.Now.Date).Substring(0, 9)));
                    command.Parameters.AddWithValue("@_every_other_day",
(edit_every_day.Checked == false) ? "T" : "");
                    command.ExecuteNonQuery();
                    connection.Close();
                }

                kk++;
                edited_number_of_intake--;
                edited_starting_time =
DateTime.Parse(edited_starting_time).AddHours(interval).ToString("HH:mm");
            }

            try
            {
                while (true)
                {
                    connection.Open();
                    command = new SqlCommand();
                    command.Connection = connection;
                    command.CommandText = string.Format("DELETE dbo.cabstorage WHERE
access_id='{1}'", edited_starting_time, access_id_retrieve[kk]);
                    command.ExecuteNonQuery();
                    connection.Close();
                    kk++;
                }
            }
            catch
            {
                MessageBox.Show("Done saving", "SAVE", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }

        }
        catch
        {
            MessageBox.Show("Nothing save, maybe you did not select a medicine name",
"SAVE", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

private void edit_terminate_Click(object sender, EventArgs e)
```

```
    {
        DialogResult result = MessageBox.Show("This action will terminate a schedule of a
medicine for the patient assigned for this drawer. Do you want to continue?", "TERMINATE",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

        if (result == DialogResult.Yes)
        {
            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;

            try
            {
                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = string.Format("DELETE FROM dbo.cabstorage WHERE
drawer_no='{0}' AND med_name='{1}'", drw_no,
edit_medicine_listBox.SelectedItem.ToString());
                command.ExecuteNonQuery();
                connection.Close();
            }
            catch
            {
                MessageBox.Show("Select first the medicine you want to terminate the
notification", "TERMINATE", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
    }

    private void edit_deallocate_Click(object sender, EventArgs e)
    {
        DialogResult result = MessageBox.Show("This action will deallocate medicine/s for this
drawer. Do you want to continue?", "DEALLOCATE", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);

        if (result == DialogResult.Yes)
        {
            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;

            try
            {
                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = string.Format("DELETE FROM dbo.cabstorage WHERE
drawer_no='{0}'", drw_no);
                command.ExecuteNonQuery();
                connection.Close();
```

```
            }
            catch
            {
                MessageBox.Show("You did not select a drawer or this drawer is empty",
"DEALLOCATE", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
    }
//############### EDIT TAB END ###################################

//##### LOG TAB ##############################################
    private void print_but_Click(object sender, EventArgs e)
    {
        printDocument1.Print();
    }

    private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
    {
        Bitmap bm = new Bitmap(this.log_dataGridView.Width, this.log_dataGridView.Height);
        log_dataGridView.DrawToBitmap(bm, new Rectangle(0, 0, this.log_dataGridView.Width,
this.log_dataGridView.Height));
        e.Graphics.DrawImage(bm, 0, 0);
    }
//########## LOG TAB END ############################

//########## INVENTORY TAB ##############################
    private void inventory_drawer_listBox_SelectedValueChanged(object sender, EventArgs e)
    {
        string drw_no2, previous = "\0";
        int total = 0;
        contents.Clear();

        if (inventory_drawer_listBox.SelectedIndex >= 0)
        {
            if (inventory_drawer_listBox.SelectedItem.ToString().Length == 8)
                drw_no2 = inventory_drawer_listBox.SelectedItem.ToString().Substring(7, 1);
            else
                drw_no2 = inventory_drawer_listBox.SelectedItem.ToString().Substring(7, 2);

            SqlConnection connection = new SqlConnection(@"Data Source=.;Initial
Catalog=C:\PROGRAM FILES\MICROSOFT SQL
SERVER\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\MEDCAB.MDF;Integrated Security=True");
            SqlCommand command;
            SqlDataReader read;

            connection.Open();
            command = new SqlCommand();
            command.Connection = connection;
            command.CommandText = string.Format("SELECT * FROM dbo.cabstorage WHERE
drawer_no='{0}' ORDER BY med_name ASC", drw_no2);
            command.ExecuteNonQuery();
            read = command.ExecuteReader();
```

```
            while (read.Read())
            {
                if (previous == read["med_name"].ToString())
                {
                    continue;
                }
                contents.AppendText("\n" + read["med_name"].ToString() + " ---------- " +
read["total_qty"].ToString() + "\n");
                total += int.Parse(read["total_qty"].ToString());
                previous = read["med_name"].ToString();
            }
            read.Close();
            connection.Close();

            contents.AppendText(string.Format("\nTOTAL: {0}", total));
        }
    }
//########## INVENTORY TAB END ######################
    private void exitToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        Close();
    }

    private void aboutToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        Form3 form3 = new Form3();
        form3.Show();
    }

}
public class MgaVar
{
    private static bool OkBut;
    public static bool okbut
    {
        get { return OkBut; }
        set { OkBut = value; }
    }
}
}

FORM2;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
```

```
namespace WindowsFormsApplication8
{
   public partial class Form2 : Form
   {
      string old_password, newpassword, recovery, activeDir, newPath;

      public Form2()
      {
         InitializeComponent();
      }

      private void Form2_Load(object sender, EventArgs e)
      {
         //This segment creates c:\password\old_password if it does not exists

         activeDir = @"c:\password";  // Specify a "currently active folder"

         if (!Directory.Exists(activeDir))
         {
            activeDir = @"c:\";

            //Combine the active directory to the new subfolder
            newPath = Path.Combine(activeDir, "password");

            // Create the subfolder
            Directory.CreateDirectory(newPath);

            // Combine the file name with the path
            newPath = Path.Combine(newPath, "old_password");

            if (!File.Exists(newPath))
            {
               // Create the file and write to it
               using (StreamWriter sw = new StreamWriter(newPath))
               {
                  //write to the file the initial password
                  sw.Write("neil");
                  sw.Close();
               }
            }
         }

         else
         {
            // Combine the file name with the path
            newPath = Path.Combine(activeDir, "old_password");

            if (!File.Exists(newPath))
            {
               // Create the file and write to it
               using (StreamWriter sw = new StreamWriter(newPath))
               {
                  //  write to the file the initial password
```

```
                sw.Write("neil");
                sw.Close();
            }
        }
    }

    //  retrieves the old_password if c:\password\old_password already exist else it will be
back to initial password
    old_password = System.IO.File.ReadAllText(newPath);

}

private void change_password_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    new_password.ResetText();
    password.ResetText();
    retype_password.ResetText();
    forgot_password.Enabled = false;
    label2.Visible = true;
    label3.Visible = true;
    new_password.Visible = true;
    retype_password.Visible = true;
    back_but.Visible = true;
    button1.Text = "CHANGE";
}

private void button1_Click(object sender, EventArgs e)
{
    if (old_password == password.Text && button1.Text == "OK")
    {
        MgaVar.okbut = true;
        this.Close();
    }
    else if (new_password.Text == retype_password.Text && button1.Text == "CHANGE")
    {
        newpassword = new_password.Text;
        if (old_password != password.Text)
        {
            MessageBox.Show("Your old password is incorrect", "PASSWORD",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        else
        {
            old_password = newpassword;
            System.IO.File.WriteAllText(@"c:\password\old_password", old_password);   //
update password
            MessageBox.Show("Password has been changed", "PASSWORD",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    else if (recovery == password.Text && button1.Text == "RESET")
    {
```

```
            System.IO.File.WriteAllText(@"c:\password\old_password", "neil");   //  update
password file
            old_password = "neil";                                          //  update password
            MessageBox.Show("Password has successfully reset", "PASSWORD",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else if (password.Text == retype_password.Text && button1.Text == "SET")
        {
            button1.Text = "RESET";
            using (StreamWriter sw = new StreamWriter(newPath))
            {
               //  write to the file the password
               sw.Write(password.Text);
               sw.Close();
            }
            //  retrieves the recovery if c:\password\recovery already exist else it will be back
to initial password
            recovery = System.IO.File.ReadAllText(@"c:\password\recovery");

            MessageBox.Show("You now have recovery password please memorize this
recovery password because you cannot change it anymore.", "PASSWORD",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        else
        {
           MessageBox.Show("Please verify your password!", "PASSWORD",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
        new_password.ResetText();
        password.ResetText();
        retype_password.ResetText();
    }

    private void back_but_Click(object sender, EventArgs e)
    {
        new_password.ResetText();
        password.ResetText();
        retype_password.ResetText();
        label1.Text = "Password";
        label2.Visible = false;
        label3.Visible = false;
        new_password.Visible = false;
        retype_password.Visible = false;
        back_but.Visible = false;
        button1.Text = "OK";
        change_password.Enabled = true;
        forgot_password.Enabled = true;
    }

    private void forgot_password_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
    {
        new_password.ResetText();
```

```
            password.ResetText();
            retype_password.ResetText();

            activeDir = @"c:\password";  // Specify a "currently active folder"


            //This segment creates c:\password\recovery_password if it does not exists
            if (!Directory.Exists(activeDir))
            {
                activeDir = @"c:\";

                //Combine the active directory to the new subfolder
                newPath = Path.Combine(activeDir, "password");

                // Create the subfolder
                Directory.CreateDirectory(newPath);

                // Combine the file name with the path
                newPath = Path.Combine(newPath, "recovery");

                if (!File.Exists(newPath))
                {
                    label1.Text = "Recovery Password";
                    button1.Text = "SET";
                    label3.Visible = true;
                    retype_password.Visible = true;
                    back_but.Visible = true;
                    change_password.Enabled = false;
                    MessageBox.Show("It seems that you do not have recovery password. Please
make one.", "PASSWORD", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    button1.Text = "RESET";
                    change_password.Enabled = false;
                    label1.Text = "Recovery Password";
                    back_but.Visible = true;
                }
            }

            else
            {
                // Combine the file name with the path
                newPath = Path.Combine(activeDir, "recovery");

                if (!File.Exists(newPath))
                {
                    label1.Text = "Recovery Password";
                    button1.Text = "SET";
                    label3.Visible = true;
                    retype_password.Visible = true;
                    back_but.Visible = true;
                    change_password.Enabled = false;
```

```
                MessageBox.Show("It seems that you do not have recovery password. Please
make one.", "PASSWORD", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                button1.Text = "RESET";
                change_password.Enabled = false;
                label1.Text = "Recovery Password";
                back_but.Visible = true;
            }
        }
    }

    private void cancel_but_Click(object sender, EventArgs e)
    {
        this.Close();
    }
  }
}
```

FORM3:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication8
{
  public partial class Form3 : Form
  {
    public Form3()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      Close();
    }
  }
}
```

E. Price List

COMPONENT	UNIT PRICE
PIC16F877A W/ HOLDER	530
MAX232	36
BUZZER	70
5mm SUPER BRIGHT LED	10
AC POWER ADAPTER	350
PCB	100
STRANDED WIRE	4/m
2 PINS TERMINAL BLOCK	12
W106 BRIDGE DIODE	10
7805 VOLTAGE REGULATOR	15
16 MHz CRYSTAL	35
HEAT SINK	20
CROWN JACK	18
SERIAL-TO-USB MALE	300
DB9 RS232 FEMALE	200
HEX INVERTER IC	5
CUSTOMIZE WOODEN CABINET	300
FIBER GLASS	125

F. Letter of Intent

Republic of the Philippines
Department of Health
TONDO MEDICAL CENTER
Balut, Tondo, Manila

Letter of Intent to Use

April 24,2012

Prof. Ayra Panganiban
Faculty
Department of Computer Engineering
Mapúa Institute of Technology
Intramuros, Manila

Dear Prof. Ayra Panganiban,

Our company, **TONDO MEDICAL CENTER** with business address at Balut, Tondo Manila, through this letter is signifying its intention to use the solution being proposed by the Computer Engineering students, **Neil Carlo P. Catalan, Josiah David D. Jose and Carla Louie H. Leandicho**, of Mapúa Institute of Technology. At present, the company is in need of a solution that will help our nurse/s in monitoring the patient's time of intake of their medicine/s. The proposed solution entitled "**Solid Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode (LED) Notifier**" was based on the aforementioned need presented to your students.

The company believes that the solution your students will provide will be of great help to our company's operation. We are looking forward for your approval of the said solution.

RHODORA D. QUILANTIP, RN, MAN
Chief Nurse
2536101

Solid-Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode Notifier

Neil Carlo P. Catalan[#1], Josiah David D. Jose[#2], Carla Louie H. Leandicho[#3]

School of Electrical Engineering, Electronics Engineering, and Computer Engineering

Mapua Institute of Technology

Muralla St., Intramuros, Manila, Philippines

[1]ncpcatalan@mymail.mapua.edu.ph

[2]jddjose@mymail.mapua.edu.ph

[3]clhleandicho@live.mapua.edu.ph

***Abstract*— The common causes of medication errors are missing doses, taking incorrect amounts and taking medicines at the wrong time. These mistakes could lead to increase discomfort, inadequate diseases prevention and possibly even death of the patient. The main purpose of this study is to lessen the medication errors and cost of the hospitals by designing a medicine cabinet. The medicine cabinet has light indicators in each drawer and an alarm that will notify a nurse if a particular patient needs medication. It has also inventory system to monitor the patient's medicine. It will be developed using a MCU (MicroController Unit), a high level programming language and a database management system. The MCU is connected to several LED (Light Emitting Diode), buzzer and to the computer. We intended to have two tests, which are the LED and alarm testing using UART terminal and the C# code**

***Keywords*—UART, MCU, LED, Medicine cabinet, Inventory system**

I. INTRODUCTION

A. Overview

Hospitals are one of the facilities that are used by people to give them medical, surgical, or psychiatric treatment and nursing care. It is important to ensure the safety and security of its patients by giving them right medication, healthy food, and clean environment. By considering safety and security, the patient will gain trust to the service of the hospital. Nurses are of big help in a hospital because they are assigned at the nurse station in each ward where they manually check the schedule of patient's time for taking their medicines as ordered by a doctor. The common problems that the nurses commit include having trouble with the patients' schedule and what medicines they need to deliver to their patients. It is critical for the nurses to give the right medicine at its scheduled time of medication to ensure the patient's safety and health.

Nowadays, hospitals are using different technologies in medication to ensure the safety of its patient. Before, hospitals are using medicine cabinet that has compartments for containing supplies of different kinds of medications to be accessed by a healthcare attendant for preparing individual medication dosages for named patients. With the use of technology, some of the medicine cabinets include a processor having a memory for storing the names of patients and their prescribed medication dosages, and a display screen for displaying the patient names and their respective prescribed medication dosages. The tray includes a display screen for displaying the patient names and their respective medication dosages, and a communication link with the medicine cabinet through which the cabinet processor communicates to the tray the patient names and their respective medication dosages. Some of the medicine cabinets have an alarm that will sound reminding the user when medicine shall be taken, and making the correct dose available. The advantages of this automated medication cabinets is that it will lower costs associated with pharmaceutical distribution, monitors inventory, further reduction of errors, and relieving professional pharmacists and nursing personnel of many tasks.

B. Customer

Our target customer is Tondo Medical Center located at North Bay Boulevard, Balut Tondo, Manila. It is a 200-bed capacity tertiary public medical center established in 1971 by virtue of Republic Act no. 6375. It presently operates under the supervision and control of the Department of Health (DOH).

Need

In a Philippine hospital usually a particular nurse is assigned at the nurse station in each ward where he/she manually checks the schedule of patient's time for taking their medicines as ordered by a doctor, and usually a group of patients is assigned to a nurse, the reason for this is to cut the cost of expenditures for nurses. This scenario is evident in government hospitals which lead to an error on the part of the nurses. The assigned nurse is the one who is responsible for preparing the medicine requirement of all the patients he/she handles in a ward (for example a medical ward). Also, the assigned nurse checks the stocks of medicines in the cabinet. With this situation, a need for a cabinet for solid medicines with time based notifier is required so that the assigned nurse will not be confused on what slot in the medicine cabinet he/she will open. If the time comes for taking a medicine, the door of the cabinet will have an LED indicator then an inventory application in a computer will check what solid medicines has to be taken out by the assigned nurse and it will check how much medicine is available, thus simplifying the work of the assigned nurse.

C. Solution

With the presented problem, the designers came with a solution to design a solid-medicine cabinet and inventory system with time based alarm and light emitting diode (LED) notifier. First it will implement an application for inventory system that will monitor what medicine should be taken out and check the quantity of available medicine. And lastly is to notify nurses on the schedule of patients on his/her medicine intake using an alarm and LED as light signals on each slot in the cabinet.

The medicine box or shelf which will be comprised of 16 drawers with 16 LEDs beside each drawer particularly at the right position, each drawer has 2" x 4 ¾" x 1 ¼" dimensions. `

D. Constraints

The constraint/s of the solution in terms of economics is first the solution requires a good budget in order to install in a particular hospital. The designers of the solution want to have a plan for actual implementation, but still the return of investment is not yet considered.

In terms of manufacturability, the solution may require a custom built cabinet for medicine and the electronic parts which may lead to larger cost compare with an ordinary cabinet. And lastly, a computer is needed for the design solution to work.

In terms of capability, the medicine cabinet is only intended to store solid medicines for oral medication. The computer is not capable of accessing other medicine cabinet. Thus, it will provide monitoring on one cabinet only.

In terms of sustainability, the medicine cabinet will use a regular adapter that is plug in to a regular outlet so sustainability in power source is not yet considered. In addition, the medicine cabinet is interfaced with the computer with database which requires maintenance by an expert.

In terms of software application, the solution will require initial inputs on the stocks of medicine, patient's name, name of the doctor, time of intake, slot number on the cabinet, the name of medicine and etc. Next, only the authorized person can assign the schedule for each nurse on the computer. Third, the connection between the cabinet and computer is not wireless. Fourth, the software application will use the system clock of the computer. And lastly the software will not recognize if the nurse has taken out the exact quantity of medicine, the quantity of medicine could be more than or less than the required number of medicine.

In terms of security, the hardware and software application provide a low level of security. Because the designers believe that security is another scope of the design, futher studies on it will not be included with the scope of this solution due to time constraints.

E. Impact

The impact of the design in terms of meeting desired needs to health and safety is that the design will help the nurses to provide the medication for the patient in proper time, and they will not be confused because the design will provide light on LEDs on the slots intended for those patients that are scheduled to intake their medicine in the cabinet.

F. Differentiation

1) Manual method: Almost all hospitals in the Philippines uses a manual method for keeping medicines and sometimes this method brings a possible cause of accident, like for example a nurse which gave wrong medicine to a patient.

Difference: Our design proposal lessens the possibility mentioned above by including a light signal; there will be a slot on the medicine cabinet for the LEDs. When the medicine needed is inside the drawer, the assigned LED for the said drawer will light up; otherwise, it will not. Moreover, the proposed system provides an organize way to remind an individual in the hospital for the patient's scheduled medication. The proposed system also keeps track of the medicines that are being stored and taken out to the patient's assign drawer, with the use of the inventory software.

2) A Medicine Box Prompter Kit Using RFID: Published on June 2008 at Mapúa Institute of Technology

The main key in this design is by the use of radio frequency in alerting the patient once he or she is out of range, which effectively reminds him or her to carry important medication before leaving home.

Difference: Our design will be implemented to a hospital particularly at the medical ward unlike the said project above it is only for a single patient that is capable of operating the apparatus to take their medication on time. It also uses RFID technology to remind the patient, while in our design we will use microcontroller to control the LED to notify the nurse for patient's medication.

3) Medication Dispensing System including Medicine Cabinet and Tray therefore:

Invented by Haitin, David and Asseo, Gilead. A medication dispensing system includes a medicine cabinet having a plurality of compartments for containing supplies of different kinds of medications to be accessed by a healthcare attendant for preparing individual medication dosages for named patients; and a tray having a plurality of sections for receiving a plurality of receptacles each adapted to contain one or more medication dosages prescribed for a named patient. The medicine cabinet includes a processor having a memory for storing the names of patients and their prescribed medication dosages, and a display screen for displaying the patient names and their respective prescribed medication dosages. The tray includes a display screen for displaying the patient names and their respective medication dosages, and a communication link with the medicine cabinet through which the cabinet processor communicates to the tray the patient names and their respective medication dosages.

Difference: Our design project can differentiated from the said invention which requires a construction of cabinet processor while our design does not need it. In our design solution we will not include tray with multiple compartments unlike the design mention above. Our design solution will use a computer for its inventory of medicine while the design mention above will use a processor having a memory for storing the names of patients and their prescribed medication dosages. So, with the manufacturability of our design much more simple than one mentioned above.

4) *Cabinet for Dispensing Medicines at Predetermined Times:*

Invented by McLaughlin,John T. A cabinet containing individual compartments each with an individual time lock for holding various doses of prescribed medicines respectively is positioned adjacent a given patient's bed. The individual time locks are programmed to open at given times during a 24 hour period at which the medicine in the corresponding compartment is to be given to the patient. A signal light advises a nurse whenever any one of the compartments is unlocked. It is thus assured that the correct dose of the correct prescribed medicine is given to the correct patient at the correct given time for that particular medicine.

Difference: Our design project can also differentiated to this design because the said invention is positioned adjacent to a given patient while our project is located at the medical ward nurse station and our design is only intended for solid medicine only. Moreover, our design does not provide a lock one mentioned above, but our design solution provides a light indicator for each slot in the cabinet. Our design solution also implements an inventory system that will keep track on what medicine is disposed from the cabinet in which an added features than the one mentioned above.

G. Benefits

The benefits of the proposed system are as follows: First, the checking of patient's medicine schedule is automated. Next, it provides safety in picking the medicines. And it will tell on what medicines was being disposed to patients. In short, it will serve as an electronic guide to the nurse thus; simplifying his/her effort which will make his/her do another activity inside the hospital related to his/her work.

II. TESTING

This chapter discusses the various tests conducted in relation to the stated objectives to determine the functionality and reliability of the created prototype.

Before the test, the researchers planned to have five trials on each drawer as well as with the buzzer. Using the UART terminal in Mikro C the researchers send an ASCII character to the COM port of the computer then eventually the microcontroller will interpret that character to a command that will turn on or turn off a particular LED on the drawer or will alarm the buzzer.

Table I shows the output action of the microcontroller to a specific data input. It will serves as the basis for determining if a data sent was successfully interpreted by the microcontroller.

TABLE I: EXPECTED ACTION BY THE MCU WITH THE CORRESPONDING INPUT DATA

Affected component	Data input	State of the component (output)
Led 1	a	On
Led 2	b	On
Led 3	c	On
Led 4	d	On
Led 5	e	On
Led 6	f	On
Led 7	g	On
Led 8	h	On
Led 9	i	On
Led 10	j	On
Led 11	k	On
Led 12	l	On
Led 13	m	On
Led 14	n	On
Led 15	o	On
Led 16	p	On
Alarm	q	On
Led 1	A	Off
Led 2	B	Off
Led 3	C	Off
Led 4	D	Off
Led 5	E	Off
Led 6	F	Off
Led 7	G	Off
Led 8	H	Off
Led 9	I	Off
Led 10	J	Off
Led 11	K	Off
Led 12	L	Off
Led 13	M	Off
Led 14	N	Off
Led 15	O	Off
Led 16	P	Off
Alarm	Q	Off

Table II and III show the collected data in testing the functionality of the LED and the buzzer of the created prototype, using the UART terminal in Mikro C. The test ensures that the researchers are successful in creating the hardware component of the design solution that has a light signal on each drawer and an alarm. With that, the researchers assumed that every command signal that was sending though serial communication has a corresponding action by the microcontroller unit.

TABLE II: TEST FOR THE SIXTEEN LED RESPONSES ON THEIR RESPECTIVE DRAWERS

	Trials					% of Success
	On	Off	On	Off	On	
1	Success	Success	Success	Success	Success	100
2	Success	Success	Success	Success	Success	100
3	Success	Success	Success	Success	Success	100
4	Success	Success	Success	Success	Success	100
5	Success	Success	Success	Success	Success	100
6	Success	Success	Success	Success	Success	100
7	Success	Success	Success	Success	Success	100
8	Success	Success	Success	Success	Success	100
9	Success	Success	Success	Success	Success	100
10	Success	Success	Success	Success	Success	100
11	Success	Success	Success	Success	Success	100
12	Success	Success	Success	Success	Success	100
13	Success	Success	Success	Success	Success	100
14	Success	Success	Success	Success	Success	100
15	Success	Success	Success	Success	Success	100
16	Success	Success	Success	Success	Success	100

TABLE III: TEST FOR BUZZER RESPONSE

Trials					% of Success
On	Off	On	Off	On	
Success	Success	Success	Success	Success	100

The percentage of success was computed based on a formula that can be express as

$$\frac{\text{Total number of success trial}}{\text{Total number of trials performed}} \times 100\% \qquad (3.1)$$

As clearly seen in Table II and III, the medicine cabinet was successfully responsded to every data input.

But during the design process another problem arises which lead to the question about the response of the microcontroller unit to the command send by the C# program. With this question, the designers choose test inputs for the application software. Since the main function of this solution is to schedule a patient's time of intake, the designers constructed Table IV

TABLE IV: TEST INPUTS FOR THE SOFTWARE APPLICATION.

	Trial 1	Trial 2	Trial 3	Trial 4
Software Application Inputs	12 am	7:30 am	12 pm	7:30 pm

Table IV represents the test inputs for the software application. The time in trial 1 and trial 3 are the two extreme input values because the software application converts the time input from 12 hour format to 24 hour format; in converting a time from 12 hour to 24 hour format it is known that adding 12 to the hour part of time is necessary for those time after noon, and those time before noon remain the same. But if the time is 12 pm it should remain the same and if the time is 12 am the

hour part must be subtracted to 12, because of this diffirent process in converting 12 am and 12 pm time the designers considered these times as the two extreme values.

Now, to represent a time between those extreme values the designer chose the time in trial 2 and trial 4. The said table is necessary so that the designers would not take every time in the clock, this table serve as a representative time inputs to the software application.

Table V and VI show the collected data in testing the functionality of the LED and the buzzer of the created prototype, using the trials in Table IV.

TABLE V: TEST FOR THE SIXTEEN LED RESPONSES ON THEIR RESPECTIVE DRAWERS WITH C# CODE

	Trials				% of Success
	1	2	3	4	
1	Success	Success	Success	Success	100
2	Success	Success	Success	Success	100
3	Success	Success	Success	Success	100
4	Success	Success	Success	Success	100
5	Success	Success	Success	Success	100
6	Success	Success	Success	Success	100
7	Success	Success	Success	Success	100
8	Success	Success	Success	Success	100
9	Success	Success	Success	Success	100
10	Success	Success	Success	Success	100
11	Success	Success	Success	Success	100
12	Success	Success	Success	Success	100
13	Success	Success	Success	Success	100
14	Success	Success	Success	Success	100
15	Success	Success	Success	Success	100
16	Success	Success	Success	Success	100

TABLE VI: TEST FOR BUZZER RESPONSE WITH C# CODE

Trials				% of Success (%)
1	2	3	4	
Success	Success	Success	Success	100

Table V and VI show the response of the light signal and the alarm with the test inputs of the software application it clearly shows that the solution responded to a particular scheduled time. Again the percentage of success was measured using equation 3.1.

III. CONCLUSIONS

The design Solid-Medicine Cabinet and Inventory System with Time Based Alarm and Light Emitting Diode (LED) Notifier was designed, developed, constructed and tested. The designed device was able to monitor the time intake of medicine of a patient.

The design provides a more accurate time of take of medicine. The alarm and LED notifies the nurse in charge that a patient needs to take medication through this the medication of every patient is monitored.

The design has its inventory system that will monitor what medicine should be taken out and check the quantity of available medicine. Through this inventory the medicine given to the patients will be on time. The nurse assigned will be notified by the inventory that a patient needs to take what kind of medicines and amount of medicine needed to be taken.

The tests that were conducted by the designers show the planned output of the device. The designed device will be helpful to the doctors and nurses to monitor patient's medication.

The researchers were able to conclude the following through testing: First, the solution responded to the signal that was sending using UART terminal and lastly the solution responded with the C# program at corresponding user's test inputs.

ACKNOWLEDGMENT

We are sincerely thankful to our adviser, Dionis Padilla, whose encouragement, guidance and support from the initial to the final level enabled us to develop an understanding of the subject.

It is a pleasure to extend gratitude to those who made this design project possible such as our parents who gave us the moral support, and our friends, Rommer Cañete and Francis Evangelista who helped us in the programming part of this design project. We also would like to make a special reference to Ms. Ayra Panganiban who is our professor in design course. Without her guidance, we could not have completed this design project. We also like to thank our design panels for giving us the necessary corrections in our documents.

Lastly, we offer our blessings to everyone who supported us in any respect during the completion of the project.

REFERENCES

[1] J. Glucksman, et al. "Programmed Medication Dispenser", 1968

[2] J. T. McLaughlin, "Cabinet for Dispensing Medicines at Predetermined Times", 1973

[3] T. A. Hicks and B. G. Hicks, "Portable Medicine Cabinet with Timer", 1981

[4] W. G. Hopkins, *Summarizing Data: Precision of Measurement*, 2000

[5] W.Aberle, M. Hofman, *Driving LEDs with a PIC Microcontroller*, 2003

[6] R. Kittredge, *Concepts for Transmitting Data from a PC to a Microcontroller*, 2003

[7] P. Kuwik, L. Thomas, M. York, D. Crump, D. Livingston and J. C. Squire, "The Smart Medical Refrigerator", 2005
[8] M. Foo, J. Chua, J. Ng, "Enhancing Medicine Adherence through Multifaceted Personalized Medicine Management", 2011